Assisted strategic monitoring on call for tender databases using natural language processing, text mining and deep learning

Oussama Ahmia

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

Université Bretagne Sud

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE BRETAGNE SUD

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

« Oussama AHMIA »

« Veille stratégique assistée sur des bases de données d'appels d'offres par traitement automatique de la langue naturelle, fouille de textes et apprentissage profond »

Thèse présentée et soutenue à Vannes, le 06/03/2020
Unité de recherche : IRISA
Thèse N° : 555

Rapporteurs avant soutenance :

Mohamed Nadif	Professeur, Université Paris Descartes
Thierry Charnois	Professeur, Université Paris 13

Composition du Jury :

Président :	Bruno Crémilleux	Professeur, Université de Caen Normandie
Examinateurs :	Alexandre Garel	Encadrant industriel, OctopusMind
	Jeanne Villaneau	MCF, Université de Bretagne Sud
	Nadir Farah	Professeur, Université d'Annaba
Dir. de thèse :	Pierre-François Marteau	Professeur, Université de Bretagne Sud
Co-dir. de thèse :	Nicolas Bechet	MCF, Université de Bretagne Sud

REMERCIEMENTS

Je tiens avant tout à remercier mes directeurs de thèse, Pierre-François Marteau et Nicolas Béchet, qui m'ont guidé tout au long de ces années faites de hauts et de bas, sans jamais perdre leur enthousiasme. Leurs précieux conseils et orientations ont été d'un grand apport pour mon travail.

Merci d'avoir cru en moi et de m'avoir poussé à aller de l'avant même dans les moments les plus difficiles.

Mes vifs remerciements vont également à Nadir Farah qui a su dès notre première rencontre canaliser mes efforts et booster mon ambition dans le domaine. C'est en fait grâce à lui que j'ai acquis le goût de la recherche, il a toute ma gratitude.

Aussi, je tiens à remercier Frédéric Oliveau pour sa patience, son soutien continu et l'intérêt qu'il a porté à mon travail. Ses vives critiques m'ont permis de me remettre sur les bons rails. Le soutien et l'écoute attentive d'Alexandre Garel, sa réactivité dans les moments de blocage ont facilité l'avancement de mon travail. Je le remercie il a été de très bon conseil et un ami.

Sans oublier toute l'équipe d'OctopusMind que je remercie pour leur soutien et disponibilité, ils ont toute ma gratitude. Les discussions riches et constructives que j'ai eu avec les membres de l'équipe expression ont été très bénéfiques, particulièrement avec Jeanne Villaneau et Sylvie Gibet.

Je remercie également mes rapporteurs, Mohamed Nadif et Thierry Charnois ainsi que les membres du jury, dont le président Bruno Crémilleux qui m'ont fait l'honneur de s'intéresser à mon travail.

Enfin merci à Papa, Maman, Nourhene, Alexandrine qui ont toujours été à mes côtés dans les moments difficiles et n'ont jamais cessé de croire en moi.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

0.1 Context

With the advent of the world wide web, public, semi-public and private procurement actors all around the world have been massively publishing procurement related documents on public platforms or websites. This created a relatively large corpus containing precious economical information. Nevertheless, accessing the right information at the right time, remains a challenge for every company. Moreover, a large number of strategic information is lying around, hidden in a vast amount of unstructured or semi-structured documents.

OctopusMind (previously Jurismarchés), is a French "SME" (Small Medium Enterprise) specialized in strategic monitoring applied to public procurement. Its main core business, is to notify its subscription based users with all the newly published public procurement projects related to their profiles [1], by providing them with all the documents related to this project.

It has lived through the digital revolution, from newspaper cuts up to present day using web crawlers fetching information from around the world, databases, search engines. With digitalization comes new opportunities and new challenges and you can take them as struggle or as possible opportunities for growth.

While starting this CIFRE [2], OctopusMind has already gone a long way using computer technologies in its business. Documents were harvested through a modern framework, put in a database, and sent via email. Indexing and querying capabilities of Search engines were thoughtfully exploited, as well as advanced heuristic rules to maintain quality and consistency of data and speed up the work of analysts. Also `j360.info` has just been released, targeting a wider audience, than the historic SEPAO offer [3]. These methods have their limitation as they rely heavily on human supervision

1. Customers profiles contains a set of criteria, example: activity field, financial information, localisation.
2. CIFRE (Industrial Agreement of Training through Research) is a PhD thesis prepared in collaboration of both a public research laboratory and a french private company.
3. Users with premium subscriptions, benefiting from a tailored alert service

and engineering to work properly. In order to address these limitations using more advanced NLP and text mining methods was mandatory.

In order to achieve such a goal, several tasks have to be addressed:

— Automatic information extraction, in order to find/extract a set of relevant information (limit date, financial information, project renewal detection, and son on) hidden in the text instead of doing it manually.

— Document recommendation, which consists of routing the relevant document to the relevant user/client.

— Document classification, in both a supervised way, by attributing the correct label to a given document according to its content and also in an unsupervised way, which allows us to explore the corpus in the hope of detecting interesting grouping which allows for example the detection of new emerging activities.

With the continuous improvement in Natural language processing techniques, many industries are becoming able to solve increasingly complex problems. As each year a major milestone is reached. 2001 was the advent of the first neural network based language modelling [1] allowing distributed representation for words that catches word semantics. In 2008 the use of Multi-task learning [2] in NLP, allowing to share parameters between models that are trained on multiple tasks, thus creating representations that are useful for many tasks. 2013 was the year where word2vec [3] was born, allowing better performance for RNN and CNN in NLP tasks. In 2014 Sutskever et al [4] introduced the use of sequence to sequence models for machine translation, achieving state of the art results. 2015 brought to us the use of attention neural networks for machine translation [5] beating all other methods at the time. 2018 was the year of pre-trained language models, only requiring unlabelled data to create the initial model. Usually these models can be used across a diverse range of NLP tasks. For instance BERT [6], was able to perform document classification, machine translation, and sequence labeling after it was fine tuned according to the target task.

0.2 Challenges and problematics

Nature of the data

Public procurement documents are created and published by public actors (municipalities, state owned companies, and so on). This documents usually comes in two types:

— Call for tenders (also known as call for bids) is a public procurement process that generates offers from competing companies in order to obtain an award of business contract for works, supply or service activity. The call for tender document contains a set of information (the description of a project, the maximum budget allocated, the different technical criteria, and so on), also each call for tender has a deadline, beyond which the the call for bids is considered unsuccessful. The public actor will then select the best offer according to a set of criteria.
— Contract award notice is the official form of notification issued by the public buyer to publicly notify the successful bidder and the award price.
— There also exists other variations of those documents: Cancellations notice, Call for registration, notice on future markets, design competitions, and so on [4]

Figure 1 gives an idea of available documents types solely on TED, the European Tender Platform.

4. For the purpose of this this thesis we will focus essentially on Call for tenders and "Contract award notice".

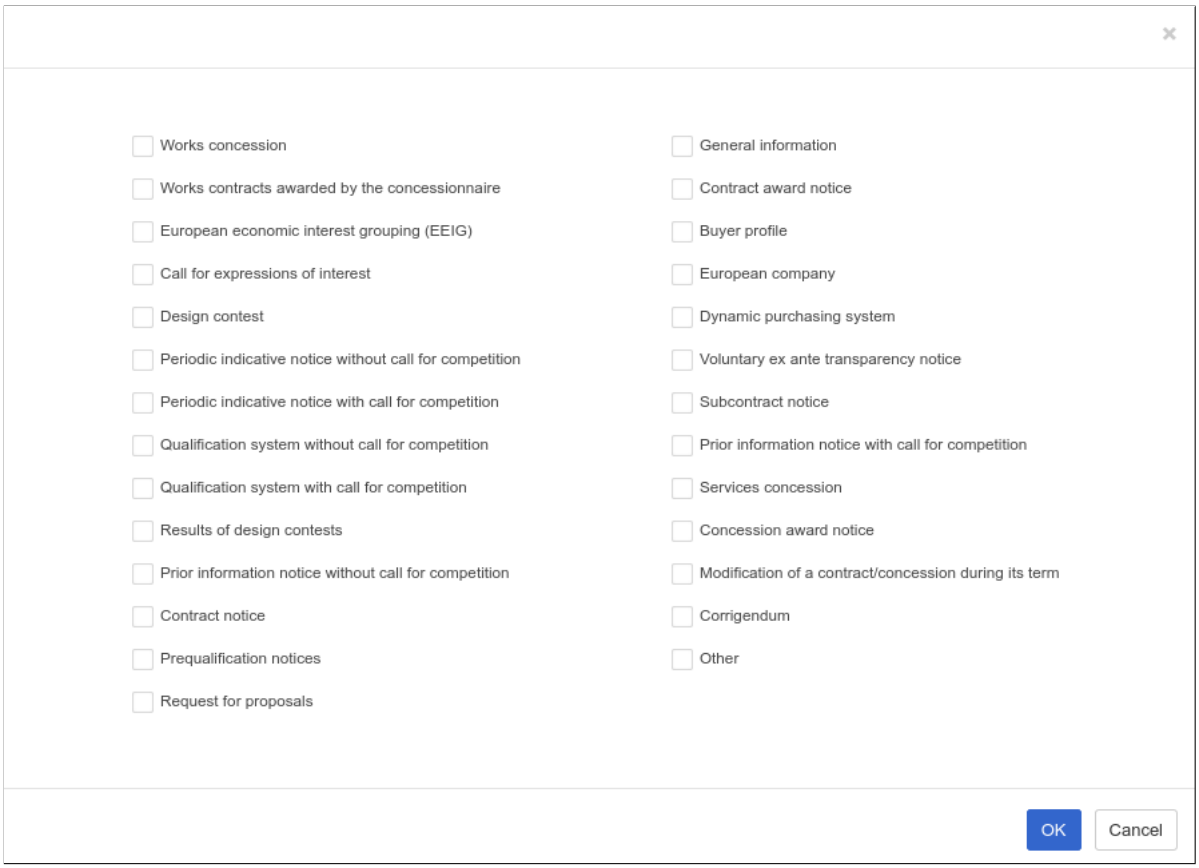

Figure 1 – Information type available on European Tender Platform TED

Each public entity around the world will publish its public procurement related documentation in the format they deem appropriate, as there is no standardized way of doing it. These format differences make it hard to process automatically documents from a source to another.

Also the fact that the documents are manually written, without accurate review, makes them prone to spelling mistake. Moreover the style can be either very telegraphic, with long sentences without verbs, or in some case overblown (in some African countries for example, where administrative contents tend to adopt a specific style).

In addition to that, the document contains administrative and legal information that covers a big portion of text. This kind of information is usually considered as noise when trying to understand the core business that describes the related project.

It is for these reasons that processing public procurement documents can be a challenging problem. Having to develop methods that mitigate the noisy nature of the data and the different format and syntactic structures from a source to another.

Hardware limitation

Developing tools with the aim of integrating them into a pre-existing software environment, comes with a set of challenges. In addition to the fact that the methods developed have to cope with some technical criteria (software compatibility, licensing type, and so on), it also has to cope with cost, thus hardware, limitations.

One of the main challenges that we take into account when developing a natural language processing model throughout this thesis, is to ensure that our solutions for the various considered tasks (system recommender, text classification, information extraction) are as efficient as possible. Therefore this efficiency objective is always taken into account when designing a model. Hence the hyper-parameters of the models we design (number of layers in a neural network for example) have to be chosen to optimize the weight/size and the response time according to the hardware on which it will be executed. Making the search for the right balance between efficiency and performance is very challenging.

Lack of related corpus

In most cases, while addressing NLP (Natural Language Processing) tasks , it is necessary to rely on textual resources to build models (for instance training an ANN or a rule-based system), whether it is for information extraction or supervised text classification. This textual resource has to rely on a domain-specific corpus that covers the task objectives.

To our knowledge, public procurement domain is rarely explored by the NLP community. Therefore, no ready-to-use related corpus were to be found.

So, creating a dataset/corpus for each task we have to tackle is an unavoidable step. However, employing experts to label a corpus can be very costly and in most cases cannot be afforded by small and average size companies. So we had to be crafty and some times tackle a problem in an indirect way to minimize the financial expenses while accepting a minimal performance cost.

Information extraction

The nature of the documents processed by OctopusMind, and the public procurement documentation in general, are usually presented in an unstructured format that contains valuable information embedded into the raw text and submerged in legal and administrative paragraphs, which make up most of the raw text.

Targeted information is used to improve the user's experience in OctopuMind's platform, also reducing the human workload. It allows the users to benefit from information that cannot be obtained otherwise, since manually processing large number of documents ranging from thousands to millions can be difficult if not impossible, and surely not cost effective.

Examples of relevant information that we aim at extracting include:

Financial data that can be indexed by a search engine allowing documents to be filtered using these criteria; Information about the awardee of a project, this information proves useful when performing competition analysis or market positioning; Evaluation criteria (especially criteria involving sustainable development or social inclusion); Various important dates in the project : deadline, renewal, etc; the place of execution; and so on .

Document embedding

In order to use Machine learning models on textual data, it is mandatory to transform the raw text into a vector of numerical features (of fixed size). This process is called document embedding (or document vectorization).

The performance of either text classification (supervised or unsupervised) or recommender system algorithms will be directly linked to the embedding method we use. Thus we had to develop a multi-purpose method to generate a document vector suited with the different NLP Tasks we needed to address. This embedding method had to satisfy a number of criteria, especially:

— Size limitation, the model has to be light weight in order to minimize the storage overhead.
— Time complexity so it can be used for real time processing;
— Also it has to store semantic information of documents allowing to reflect the semantic similarity between documents.

Recommender system

Document recommendation is the core business of OctopusMind, by rooting the most relevant information to a given user according to their domain of activity. This task is particularly strenuous due to the nature of the document we are working with. Furthermore, some of noise contained in the text can be confused with terms that characterize

the domain of activity, depending on the context in which it appears. For example the two sentences:

— "Section III: Legal, economic, financial and technical information."

— "...an existing legal framework to specifically analyze the mandate..."

The two sentences contain the word "legal": in the first sentence the word legal is in a section tile, while in the second sentence it is part of the project description.

We also have to cope with the fact that the projects, subject of the call for tender, are characterized with a deadline after which the document can no longer be recommended.

0.3 Main contributions

This thesis is divided into three chapters. The first chapter describes a multi-lingual corpus, that has been made available for the NLP community. The second chapter proposes a survey of the state of the art in documents embedding and describes our contribution to this field. The last part focuses on the different solutions we have developed and deployed in a production environment.

Two multi-lingual parallel corpora

In the chapter 1.1 we describe our first contribution, which takes the form of a multi-lingual corpora "Tenders Electronic Daily" (TED) composed of two sub-datasets[7] created from the TED's [5] documents. We start by describing TED platform, by giving some quantitative statistics and some use cases for which the database can be used.

The first sub-dataset, called fd-TED, is a (multilingual) corpus or aligned translated documents contains about 3 million documents translated into the 24 languages of the European union (DA, DE, EN, ES, FI, FR, EL, IT, NL, PT, SV, CS, ET, HU, LT, LV, MT, PL, SK, SL, GA, BG, RO, HR). Each document is labeled with a set of hierarchical classes designating the core business of the project described in the document. We also provide for each document a filtered version that is stripped of all the legal and administrative content. We describe the process we got through to produce this filtered version. This dataset can be used as a benchmark for supervised document classification or to train machine learning models for business intelligence application.

5. https://ted.europa.eu/

The second sub-dataset, called par-TED, is a corpus of aligned sentences translated to 24 languages extracted from the fd-TED corpus. It can be used for machine assisted translation of juridical and technical documents, or the extraction of multilingual terminology. This corpus contains 4 millions of unique sentences translated to at least 23 languages. We explain in the related chapter, the heuristics we used to generate the par-TED dataset.

Document embedding

In the chapter 2, we address the document embedding problem. We start by a survey on the state of the art methods for document embedding. We then present our contribution and position it within the state of the art, we also explain the thinking process that drove us to develop such models. At the end of this chapter, we present the experiments we carried out in order to evaluate the performances of our contribution in comparison to the similar state of the art methods. Our contribution in this field comes in this two following models:

Combining Latent Semantic Analysis and Word2vec

The first model referred, to as LSA+W2V [8], based on the combination Word2vec and LSA vectorizations. This model takes advantage of these two well established lexical semantics methods, thus allowing us to produce a low dimensional document vector that catches both the general semantics of a document (general meaning) and maintaining a lexical-conceptual description of this document.

Convolution based neural network for efficient text classification

The second model is a novel neural network architecture for text classification that will be referred to as CnHAtt. Based on attention mechanism[9]. It is characterized by a hierarchical structure allowing the different sentences to be separately considered according to their importance. It enables large documents to be processed as it takes into account the hierarchical nature of the text. The attention is calculated using a convolution[10] and a pooling layers. We made the choice of not relying on a Recurrent Neural Network used in traditional models[11] [9], which allows it to be easily parallelisable, therefore decreasing significantly the training and prediction time. We also developed a number of variants, that gives more flexibility regarding the generated document vector.

We also developed a number of variants, that gives more flexibility regarding the

generated document vector.

Applicative tasks

Chapter 3 focuses on the Applicative contributions, as we will describe the solutions we developed and deployed to solve practical company problems, by explaining the different steps to solve each problem (data preparation, method design, evaluation). This chapter is split up into two parts that address information extraction and recommender system.

In the first part of this chapter, we describe the information extraction problems we have solved. We start with a survey of the state of the art methods, then we explain the motivation behind the models we chose to solve our problems, namely surface extraction, market renewal detection and financial data extraction.

The second part of this chapter is about the creation of a recommender system. We start survey of the state of the art methods to address this problem. Then we explain the different models we used and we access their performance.

CHAPTER 1

TED

1.1 Introduction

Before tackling any NLP (Natural Language Processing) task, whether for information extraction or supervised text classification, it is mandatory to rely on a domain-specific corpus that covers the task objective.

One of the most important task at OctopusMind is related to automatic document categorization. Being able to assign categories to documents according to their content is a priority, as such ubiquitous task allows for the automatization of a wide range of services, ranging from automatic recommender system (aiming at routing relevant documents to the right user) to decision support tools, decreasing the experts/human workload, hence saving time and costs.

In addition, document categorization allows to handle the deluge of data, since being able to automatically process thousands of documents each day is impossible to achieve manually or at least too costly for small business companies to afford.

Before starting any large scale classification task, as stated earlier, building a corpus dedicated to this purpose was a priority. The corpus had to be multilingual (as OctopusMind process document worldwide). It also need to be related to public procurement and documents may embed some labels or labelling indications allowing to use such corpora for the training of supervised classification models.

The "Tenders Electronic Daily" (TED) is the platform used by the European union government to publish their public procurement related projects. Among other similar sources, the TED platform publishes approximately 1,700 tenders, five times a week, on the TED servers [1], or 460,000 contract notices per year. Among which 175,000 tenders describes projects worth around 420 billion euros.

The TED is a large source of semi-structured and multilingual data. This dataset can

1. http://ted.europa.eu/TED/main/HomePage.do

effectively be used to address complex machine translation, multilingual terminology extraction, text-mining, or to benchmark information retrieval systems. In addition, for each document (call for tenders, award notification..) a set of labels is assigned to precisely describe the core business information that describes the related project.

These raw data are available as bulk downloads in XML format with a complex structure. Unfortunately, different versions of the XML data structure from year to year have been used, making the aggregation of the different bulks of data difficult. Furthermore, the collected documents are associated to a variable number of translations as well as variable sets of meta data that is used for indexing.

Despite the services offered by the user-friendliness of the web site that made available to the public the access to the EU call for tenders publications, collecting and managing such kind of data is a great burden and consumes a lot of time and computing resources. This could explain why such a resource is not very (if any) exploited today by computer scientists or engineers in NLP.

In order to create our corpus, we decided to process the TED database as it satisfies all of our previously listed needs.

The aim of this chapter is to describe the process we got through, in order to produce a processed version of this database, in a raw text format that can be directly and easily used for text mining and other natural language processing tasks. The corpus is decomposed into two documented and easy-to-use multilingual corpora (one of them is a parallel corpus), extracted from the TED web source that we will use to train different models to solve our NLP tasks at OctopusMind. The corpus is also made available to the scientific community and can be downloaded [2] along with a simple Python API for easier manipulation and few code samples for text classification.

The provided dataset is composed of two sub-datasets created from the TED's documents that have been published between January 2011 and August 2017 [3].

1. The first sub-dataset, fd-TED, is a (multilingual) corpus or aligned translated documents. It contains around 3 million of documents translated to 24 languages (DA, DE, EN, ES, FI, FR, EL, IT, NL, PT, SV, CS, ET, HU, LT, LV, MT, PL, SK,

2. `https://github.com/oussamaahmia/TED-dataset`

3. This numbers and statistics describes the initial upload. By the time this thesis is written we are adding the documents published until August 2019.

SL, GA, BG, RO, HR). This dataset can be used as a benchmark for supervised classification or for training machine learning models applied to business intelligence applications.

2. The second sub-dataset, par-TED, consists of the aligned sentences of translated texts extracted from the fd-TED corpus. It can be used for machine assisted translation of juridical and technical documents, or the extraction of multilingual terminology. This corpus is composed of 4 million unique sentences translated to at least 23 languages.

The two sub-datasets, fd-TED and par-TED, will be updated in the future in a regular basis to keep tracks of the new calls for tender published by the EU states.

We will also provide an API, to download the new updates and to support an easy access to the data. This is done through the use of filters that can be applied on the meta data, basically the language(s), the hierarchical level(s) of the Common Procurement Vocabulary (CPV) codes, the type of processed texts, and so on.

1.2 The (full-document) fd-TED corpus

The fd-TED corpus is built from the full content of the documents extracted from the TED platform. Each document of the corpus belongs to a hierarchy that is succinctly described below.

1.2.1 Common Procurement Vocabulary

Common Procurement Vocabulary (CPV) [4] is the thesaurus that defines the subject matter of public contracts, allowing companies to easily find public procurement notices according to their areas of expertise. The main CPV vocabulary is based on a hierarchical structure (a tree structure) comprising codes of up to 9 digits (the ninth digit serves to check the previous digits). The CPV code consists of 8 digits that encodes 5 hierarchical subdivisions as follows:

1. The first two digits identify the divisions (XX000000-Y), e.g. "industrial machinery".

4. COMMISSION REGULATION (EC) No 213/2008 of 28 November 2007

2. The first three digits identify the groups (XXX00000-Y), e.g. "Machine tools".
3. The first four digits identify the classes (XXXX0000-Y), e.g. Metal-working machine tools.
4. The first five digits identify the categories (XXXXX000-Y), e.g. "Hydraulic presses".
5. Each of the last three digits gives a greater degree of precision within each category.

For Example:
42000000 is the code for "industrial machinery", 42600000 is for "Machine tools", 42630000 for "Metal-working machine tools" and 4263600 is for "Hydraulic presses".

Table 1.1, presents the number of documents for each level of the CPV codes by taking into account the last hierarchical level (the 8 digits of the CPV code)

Level in the hierarchy	1	2	3	4	5
Count	1,868,420	433,111	231,167	144,393	115,487
Level in the hierarchy	6	7	8	9	
Count	45,656	30,792	21,694	16,727	
Level in the hierarchy	1	2	3	4	5
Cumulative	2,907,447	1,039,027	605,916	374,749	230,356
Level in the hierarchy	6	7	8	9	
Cumulative	114,869	69,213	38,421	16,727	

Table 1.1 – Number of documents for each level of the CPV code

1.2.2 The documents

The documents are published in 24 languages of the EU. They can be fully translated to the 24 languages (Table 1.2 and 1.2.2) or partially translated (in most of the cases the object of the document and the lots [5] are translated).

The dataset that we provide is presented as a multilingual corpus that can be exploited for supervised hierarchical classification or Cross-Language Text Classification [12].

5. Tenders are generally advertised with a global title (object) and some of them are divided into lots, each having its own title

Language	DE-ES	DE-IT	EN-DE	EN-ES	EN-FR
Count	425,797	428,097	425,893	425,808	426,027
Language	EN-IT	FR-DE	FR-ES	FR-IT	IT-ES
Count	425,856	429,039	425,797	425,803	425,797

Table 1.2 – Number of documents fully translated for some pairs of languages.

SL	SK	DE
451.1K/1.2M/433.0K	452.0K/1.4M/491.6K	849.4K/6.4M/1.5M
LT	GA	PT
457.3K/1.6M/496.3K	425.8K/868.6K/359.9K	450.7K/1.0M/371.1K
MT	SV	LV
425.8K/913.8K/352.1K	499.8K/1.3M/592.8K	443.3K/1.1M/420.9K
EL	FI	HU
461.5K/1.7M/526.7K	472.2K/1.3M/696.6K	457.4K/2.5M/691.8K
EN	DA	FR
674.8K/4.2M/824.9K	461.3K/1.4M/545.6K	1.1M/11.4M/1.2M
RO	PL	HR
483.1K/3.5M/567.8K	739.8K/11.2M/988.6K	288.3K/765.3K/314.9K
NL	CS	ET
525.1K/2.0M/613.5K	527.3K/1.8M/563.4K	441.5K/1.1M/510.0K
ES	IT	BG
560.3K/2.1M/510.0K	544.9K/2.9M/677.3K	485.3K/2.4M/540.6K

Table 1.3 – Number of (fully translated documents/ unique sentences/ unique words) per language.

The XML schema comes in different versions (R2.0.9 and R2.0.8), hence the needed fields are extracted using the parser corresponding to each version. Then the CPV codes are corrected if additional characters are found.

The raw text is created for each available language by converting the XML into text records. In case of any error in this step, the text is downloaded directly from the TED's website or converted using TED's online API.

Knowing that the procurement notices contain legal and administrative information that are not fundamental for understanding the core business of the consultation, as filtered sentences tends to introduce a lot of noise if the interest is upon valuable business information present in call for tenders (conditions relating to the contract, deposits and guarantees required...).

With the help of experts in public markets (hired by OctopusMind), we provide a filtered version of each document that only contains the description of the expected supplies. This consists in a filtered project description of a document in which the legal/administrative sentences have been removed, as the administrative content is considered as noise and do not contribute any useful information when predicting CPV classes.

Example of core business information:

— Installation of doors and windows and related components.

Example of legal and administrative information that has been filtered out:

— Candidates (all partners in the case of a consortium) shall prove that they have the legal capacity to perform the contract by providing (...)

The filtered descriptions fields (named "desc") are created from the aggregation of several XML elements that are checked for administrative sentences using a classifier trained with manually tagged dataset by OctopusMind [6] experts.

The dataset used for legal/administrative sentences detection was created as follow:

We started by asking the experts to list the sections of a document that will always contain administrative or legal text, then we had to find the XML path related to this section. We consider all the sentences from this section as administrative text.

For the non administrative sentences (core sentences) we keep all the document titles, and considered them as core content.

We use this dataset to train a tf-idf based Random Forest [13]. The performances

6. https://www.octopusmind.info/

of the model are shown in 1.4, using a train/test split of 80% and 20% respectively.

The filtered text is created by ignoring all the XML entities dealing with administrative information (some XML elements will always contain only administrative content) and filtering the mixed elements using the classifier to get rid of the administrative content.

	Precision	Recall	F1-score
Core	0.99	0.99	0.99
Administrative	1.00	1.00	1.00

Table 1.4 – Results obtained for administrative sentences detection.

The data structure of the documents contained in the fd-TED corpus is presented in Figure 1.1.

```
{"ref":0000-0000 #The document ID in the TED database.
 "origin_ln":"" #The original language of the document.
 "list_ln": [] #the list of languages in which the document is translated.

 "document":{
     "EN":
         {"title":"Document Title" #The title of the document.
          "CPV":['00000000'] #The list of CPVs codes of the document
          "desc": "description of the project"
                       #additional information about the project.
          "lots": [ #list of the parts of the project.
                    "title":"Title of the lot"
                    "CPV": ['00000000'] #the CPV codes of the lot.
                    "desc": "description of the lot"
                         #additional information about the lot.
          ]
          "raw": "the raw text of the document" # full text
          "filtered": "the processed document" # the text without the
          administrative information.

         }
 }}
```

Figure 1.1 – Format of the documents for the processed dataset fd-TED.

It is likely that the proposed format of the processed document will change in the near future for better storage efficiency. However the information stored will be the same.

1.2.3 Classification Example

As an example of supervised classification, Table 1.5 shows the results of a classification using Linear Support Vector Machine (SVM) [14] and a bag of words representation. From a random sub-sample of 200K English and French documents extracted from the fd-TED corpus [7], we randomly split our data into into 75% for training and 25% for testing. We have used for this experiment the first hierarchical levels of the CPV codes, namely the two first digits.

The combination of the model trained on the English version of the documents and the French one using a *max* rule [15] increases significantly the accuracy of this classification task.

Language	Accuracy
FR	59%
EN	65%
EN+FR	68%

Table 1.5 – SVM classification results

In chapter 2, we will address in more details the state of the art in text classification, specifically related on word, phrases and document embedding for text classification and will detail our contribution in this domain.

1.3 The (parallel) par-TED corpus

Alongside with the fd-TED corpus, we provide a multilingual aligned corpus in the form of a set of parallel sentences with at least 1.2 million unique sentences translated to at least 23 languages [8]. This corpus is created by aligning the XML trees for each lan-

7. We use the raw version of fd-Ted

8. This numbers and statistics describe the initial upload. By the time this thesis is written we are adding the documents published until August 2019.

guage. Some XML elements are ignored (such as Phone numbers, email, addresses, etc). Then the repeated sentences are deleted.

Below is an example of aligned sentences for the EN,FR,ES,and IT languages.

— FR: "Travaux de finition et de rénovation pour le complexe tokamak, le bâtiment d'assemblage et tous les bâtiments voisins."

— EN: "Finishing and retrofit works for the Tokamak complex, assembly hall and all surrounding buildings."

— ES: "Obras de modernización y finalización del complejo y taller de montaje del Tokamak y de los edificios colindantes."

— IT: "Lavori di rifinitura e di ammodernamento per il complesso Tokamak, il reparto di assemblaggio e tutti gli edifici circostanti."

The data structure for the par-TED corpus is presented in Figure 1.2.

```
{"ref":0000-0000 #The document ID in the TED database.
 "origin_ln':"" #The language of the source document.
 "sent_id": #Sentence id in the document.

 "sentences":{ #List of the translations.
     "EN":"...",
     "FR":"...",
     "ES":"...",
     ...
          }
 }}
```

Figure 1.2 – Format of the documents in the par-TED corpus.

As an example, we have built word embedding for the EN and FR languages to show the potentiality of this corpus in a multilingual terminology extraction application.

From Table 1.6 to Table 1.8 we can see that using a cosine similarity on Word2Vec representations [3] built on this corpus, we get comparable results regarding the word similarity on excerpts of common and proper nouns for the two tested languages.

Word(EN)	Similar Words(EN)	Similarity(EN)	Word(FR)	Similar Words(FR)	Similarity(FR)
Twitter	facebook	0.85	Twitter	facebook	0.90
	social media	0.82		instagram	0.86
	blogs	0.78		netflix	0.84
	web chat	0.69		snapchat	0.82
	press releases	0.68		google	0.81
	youtube	0.67		tweets	0.80
	newsletter	0.66		youtube	0.80
	text messaging	0.65		linkedin	0.77
	direct mail	0.65		maddyness	0.77
	google	0.65		tweet	0.77

Table 1.6 – Example of some most similar words to the word "Twitter" on the English and French Corpus using word2vec representation and cosine similarity.

Word(EN)	Similar Words(EN)	Similarity(EN)	Word(FR)	Similar Words(FR)	Similarity(FR)
Linux	citrix	0.81	Linux	windows	0.85
	unix	0.81		redhat	0.83
	server	0.80		unix	0.83
	vmware	0.80		mac os	0.82
	microsoft	0.78		citrix	0.81
	windows server	0.77		serveurs	0.80
	weblogic	0.77		microsoft	0.79
	oracle	0.77		ibm	0.79
	ms sql	0.77		windows server	0.79
	red hat	0.77		env windows	0.79

Table 1.7 – Example of some most similar words to the word "Linux" on the English and French Corpus using word2vec representation and cosine similarity.

Word (EN)	Similar Words (EN)	Similarity (EN)	Word (FR)	Similar Words (FR)	Similarity (FR)
	legal	0.70		representation	0.75
	matters	0.69		conseils	0.74
	legal matters	0.68		droit social	0.74
	advisers	0.66		assistance juridique	0.73
lawyer	disputes	0.66	avocat	avocats	0.73
+	matters arising	0.65	+	conseils juridique	0.72
advice	legal advice	0.65	conseil	representation juridique	0.72
	specific issues	0.65		contentieux	0.72
	lawyers	0.65		representation devant	0.71
	advice guidance	0.65		conseil juridique	0.71

Table 1.8 – Example of some most similar words to the sum of the word vectors "lawyer + advice" and "avocat + conseil" on the English and French Corpus using word2vec representation and cosine similarity.

We also note that the concept to which a word is related, may vary from a language to another. For example in the English version the word "Twitter" is related to words like "press releases", "newsletter", but also to words like "facebook" and "social media" . Which indicates that in the English version, the word "Twitter" is considered to belong to two concepts: press / journalism and "social media". While in the French version, the word "Twitter" is more of a "social media" concept as it is closely related to the words facebook, instagram, and so on.

1.4 Conclusion

The "Tenders Electronic Daily" (TED) is a large source of semi-structured and multilingual document widely under-used by the NLP community, mainly due to the burden and costs associated to the collecting and formatting of the data. To ease the exploitation of this resource either for text mining or machine translation tasks, we have presented a packaging of these data (along with a Python API to access it) that can be freely downloaded [9] and used by scientists and engineers to benchmark or solve some of their NLP problems. More practical applications examples will be also detailed in the chapter 3 to highlight the usefulness of this this resource and the type of services it can provide.

9. https://github.com/oussamaahmia/TED-dataset

CHAPTER 2

DOCUMENT EMBEDDING

2.1 Introduction

With the constant growth of online information, the need for developing methodologies and tools for finding, filtering and managing these resources in a fast and efficient way is an ever demanding necessity. The classification of textual data consists of classifying texts automatically in one or more categories. It has already been applied to several issues including information retrieval tasks [16], sentient analysis [17], Spam filtering [18], etc.

In order to apply machine learning algorithms (i.e. classification algorithms), in particular deep neural networks, on textual data, it is necessary in most cases to transform the texts into a fixed-size vector, in order to map the data into a metric space, this process called document embedding (text vectorisation). Numerous "vectorization" methods have been developed over the years.

Document embedding or document vectorisation can be seen as a feature extraction, that allows to transform a raw text into a mathematical representation (numeric features) that can be used by different algorithms and models. It is a very sensitive step as it garbles the original text, by creating representations that usually simplifies the nature of a document by focusing on some characteristics and omitting others (for example: ignoring sequentially in sentences). Therefore, by simplifying a document's characteristics we might end up with the ideal simplified representation in best cases and, in worst cases, with a representation that drops all useful aspects of a text. We also note that the performance of a specific document embedding approach may vary from a task to another and also according to the corpus used, as shown later in this chapter.

At OctopusMind document embedding is at the heart of several tasks, namely:

— Document classification, by assigning to each document custom labels that helps indexing and organizing documents.

— Recommendations system, that allows choosing which document to show to which client. Based on semantic distances of the new documents and the previously seen documents.
— Clustering of documents, allowing to detect emerging professional fields.
— Search engine, that returns results semantically close to the query.

The objective for OctopusMind is to develop a polyvalent document embedding model such as to ensure that the previous tasks are the more efficient as possible. Also, our vectorization model has to be fast and lightweight to cope with the hardware available at OctopusMind.

This chapter focuses on our contributions on document embedding for text classification.

We start by stating the literature and state of the art on document embedding in the Related works section. Then we introduce our contribution and position it within the state of the art. We continue by explaining the different experiments carried out to assess the performances of our contributions. Finally we list the future works and perspectives that may improve our work.

2.2 Related works

Central to many natural language processing problems, documents embedding (or documents vectorization) is about document representation such as to capture "meaning" of a document in a machine-understandable format (usually in the form of a fixed size vector). Many different vectorization approaches have been developed over the years.

The first pioneering method, which is still the most widely used, is called *Bag Of Words* (BOW) [19], [20] , which consists in describing a text using the number of occurrences of the words (i.e. their frequencies) that compose it.

Usually BOW is implemented as a sparse vector [1] of N dimensions, where N is the size of the vocabulary.

Using bag of words, the length of the vector is the size of the vocabulary and each column (feature) is a word. For example the bag of word representation for the sentence

1. BagOfWords can also be binary. In this case the frequencies are replaced by binary values that mark the presence or absence of words as a Boolean value, 0 for absent, 1 for present.

"the white cat" would be as follows: "with the given features (words) *brown, cat, dog, I, is, love, the, white*"

$$\begin{bmatrix} brown & cat & dog & i & is & love & the & white \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{2.1}$$

If we put apart the frequency of a term, this method considers that all the words in a given document to have the same weight as long as they have the same frequency, which is problematic for poorly discriminating words (stop words for example). And also BOW does not take into into account the occurrence of a word across documents.

In order to overcome this problem, Karen Spärck Jones introduced the concept of *tf-idf* weighting [21] short for (term frequency–inverse document frequency). It consists in giving more weight for the words that appear in fewer documents. Thus, if two words w_1 and w_2 have the same frequency in a given document, if w_1 appears in fewer documents than w_2, w_1 will have more weight than w_2.

This is done by assigning to each term a score based on its "term frequency"(*tf*) and on the "inverse document frequency" (*idf*). It is calculated as follows:

$$\begin{aligned} tf_{t,d} &= \frac{f_{t,d}}{\sum_k f_{k,d}} \\ idf_i &= log\frac{N}{n_k} \end{aligned} \tag{2.2}$$

Where $f_{t,d}$ if the number of occurrences of the term t in the document d, N is the number of documents and n_k is the number of documents that contains the term k. The terms with higher weights are considered to be more important.

Bag of words and *tf-idf* representations generate an embedding in very large vector space (very high dimensionality) and do not take into account the semantics nor the words order(words sequentiality). This means, for example, that the words "feed", "eat" and "drive" assuming that they have the same frequencies are equidistant in this embedding space despite the fact that semantically speaking "eating" should be closer to "feeding" than "driving".

The matrices resulting from a BOW representation with a tf-idf weighting are usually called term-document matrix or document-term matrix.

Assuming that words used in the same contexts tend to have a similar meaning, Scott Deerwester et al. introduced the concept of latent semantic analysis (LSA) [22]. LSA is a statistical technique for extracting patterns in a form of relationships between terms and concepts in documents by applying a singular value decomposition (SVD) on a term-document matrix. Furthermore, this matrix factorization technique greatly reduces the dimensionality (the size of the "embedding space"/"feature set"). LSA will be explained in more details later on in this section.

Since the advent of word2vec, introduced by Mikolov [23], new opportunities have recently appeared in the field of natural language processing and especially in document embedding. Word2vec is a word level embedding method that generates dense features vectors in a way that words with a similar meaning will have a similar representation. This method will also be explained in details later on in this section. We note that, in addition to word2vec, other word embedding models have been developed. We can mention in particular the GloVe[24] (Global Vectors for Word Representation) model. Unlike word2vec that relies on local information (lexical semantics based on a word and its surrounding words), GloVe captures both global and local statistics of the training corpus by creating word embedding based on a co-occurrence matrix.

Many attempts have been made to combine word2vec outputs with other kind or level of vectorization. Based on word2vec, doc2vec [25] provides a vector representation at a document level. The idea behind doc2vec is to represent each word with a one hot vector (binary bag of words) as in word2vec and in addition, each document is also represented as a one hot vector and is considered as a special token. Doc2vec comes in two different architectures. Both of them are inspired by learning word vectors which are skip-gram and continuous bag-of-words (CBOW). The first one is PV-DM (Distributed Memory Model of Paragraph Vectors), in this architecture the document vectors and word vectors (a sliding window of words over the document) are both represented by dense vectors that are initialized randomly. Each document in the corpus have a different vector. The document vector and the word vector are then concatenated together to predict the next word as shown in the figure 2.1. The document vector and word vector are trained during this process.

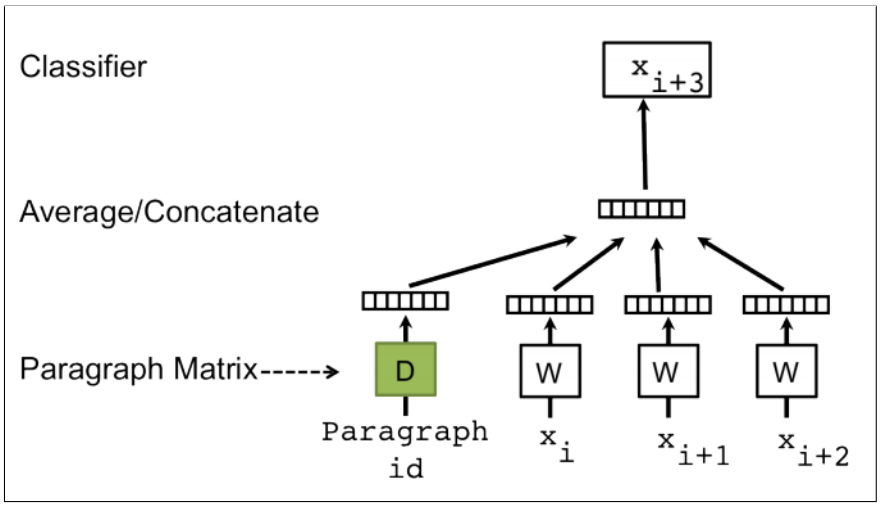

Figure 2.1 – Distributed Memory Model of Paragraph Vectors architecture taken from [26]

The second architecture is the PV-DBOW (Distributed Bag of Words version of Paragraph Vector), in this architecture the paragraph vector is used to predict randomly sampled words in the document. The classification task is to predict whether or not a word belongs to the document, as shown in figure 2.2. This is done by using the document vector to predict a randomly selected word from a randomly sampled window of the document. The vector of the word is learnt this way alongside the document vector.

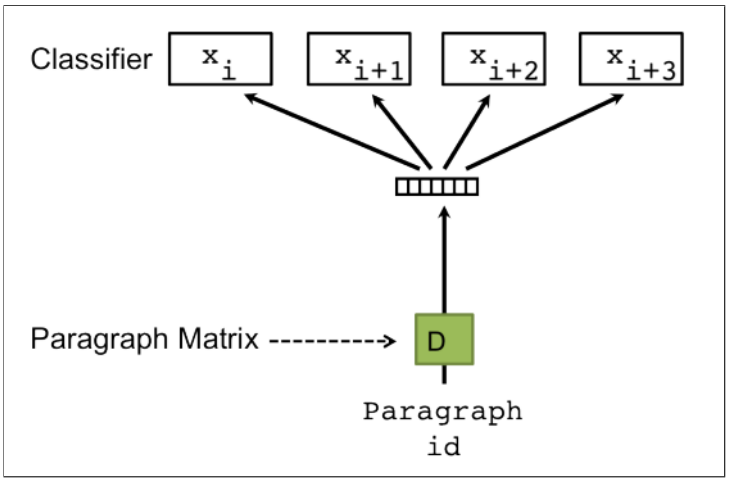

Figure 2.2 – Distributed Bag of Words version of Paragraph Vector architecture taken from [26]

In both architectures, after training, the paragraph vector is used as a document representation.

Although doc2vec has produced promising results according to [25], [26], these results could not be exactly reproduced according to [27], [28]. Furthermore, this approach requires clean data (with few stop words and non important word/sentences in general) since it gives the same importance to each word occurrence when training document vectors [29]. It also needs also a very large number of iterations to converge. The complexity of doc2vec depends on the size of the vocabulary [30] (the number of words in the vocabulary), it is expressed as $(O(log(N))$ where N is the vocabulary size.

Another interesting approach was proposed by Ronghui Ju et al [31] that combines a vectorization based on LSA, with tf-idf weighting, and a word2vec vectorization.

The approach consists in creating a $(m * n * v)$ matrix where m is the vocabulary size, n the number of documents and v the word2vec vector dimension. In this three-dimensional matrix, each document is represented as a matrix where each line is a word2vec vector weighted by tfidf (this is done by a point-wise multiplication between word2vec vectors and tfidf weight of the corresponding word).

We end up with a tri-dimensional matrix that associates the word2vec vectors of the terms with the documents. A Singular value decomposition (SVD) is then applied on on each term by document 2-dimensional matrix)

The result is then used to train a convolutional neural network (CNN) [32] connected to a pooling layer and used as an encoder (CNN will be developed in section 2.2.4). The document vectors are outputted by the pooling layer of the CNN. We note that the document vectors depend on a classification task. The model steps are shown in the figure 2.3.

According to [31], this model seems to perform better than LSA associated to a weighted bag of words vectorization. However the results presented on the paper are based on a very small dataset. Also the method is memory greedy (for example if we want to apply this approach on all the classes of 20NewsGroups we would need to store a matrix of $(20,000 \times V \times D)$ where V is the number of unique words in the corpus and D is word2vec dimension).

LDA2vec is also a hybrid model resulting from the combination of word2vec with LDA [33] (Latent Dirichlet Allocation) which is a topic modeling algorithm. This model is developed by Christopher Moody [34] and is inspired by the skip-gram variant of word2vec.

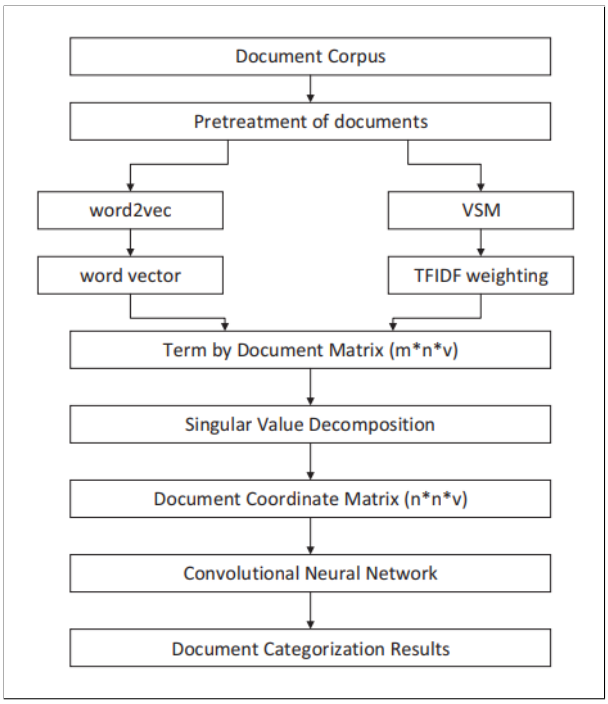

Figure 2.3 – LSA + word2vec architecture taken from [31].

The basic idea behind this model is to map word vectors and document vectors in the same embedding space. As shown in figure 2.4, the model starts by initializing a document weight matrix $D[documents \times topics]$, a topic matrix $T[hidden\ units \times topics]$ and a word matrix $W[words \times hidden\ units]$. A *softmax* function is applied to the document weight vector D_i (i is i-th document line in the matrix D) resulting in a proportion vector (similarly to LDA, a proportion vector can be interpreted as a probability distribution of topic mixtures) that sums to 1. The document vector V is then obtained by a weighted sum of the proportion vector and the topic matrix T. The central word W_p (p is p-th line in the W matrix) of a sliding window of words over the document is considered as a pivot, the word vector W_p is then summed to document vector V_i to get what the author describes as a context vector $C_{ip} = V_i + W_p$. The context vector is then used to predict a randomly selected target word drawn from the window containing neighboring words of the pivot word. The overall process is described in figure 2.4.

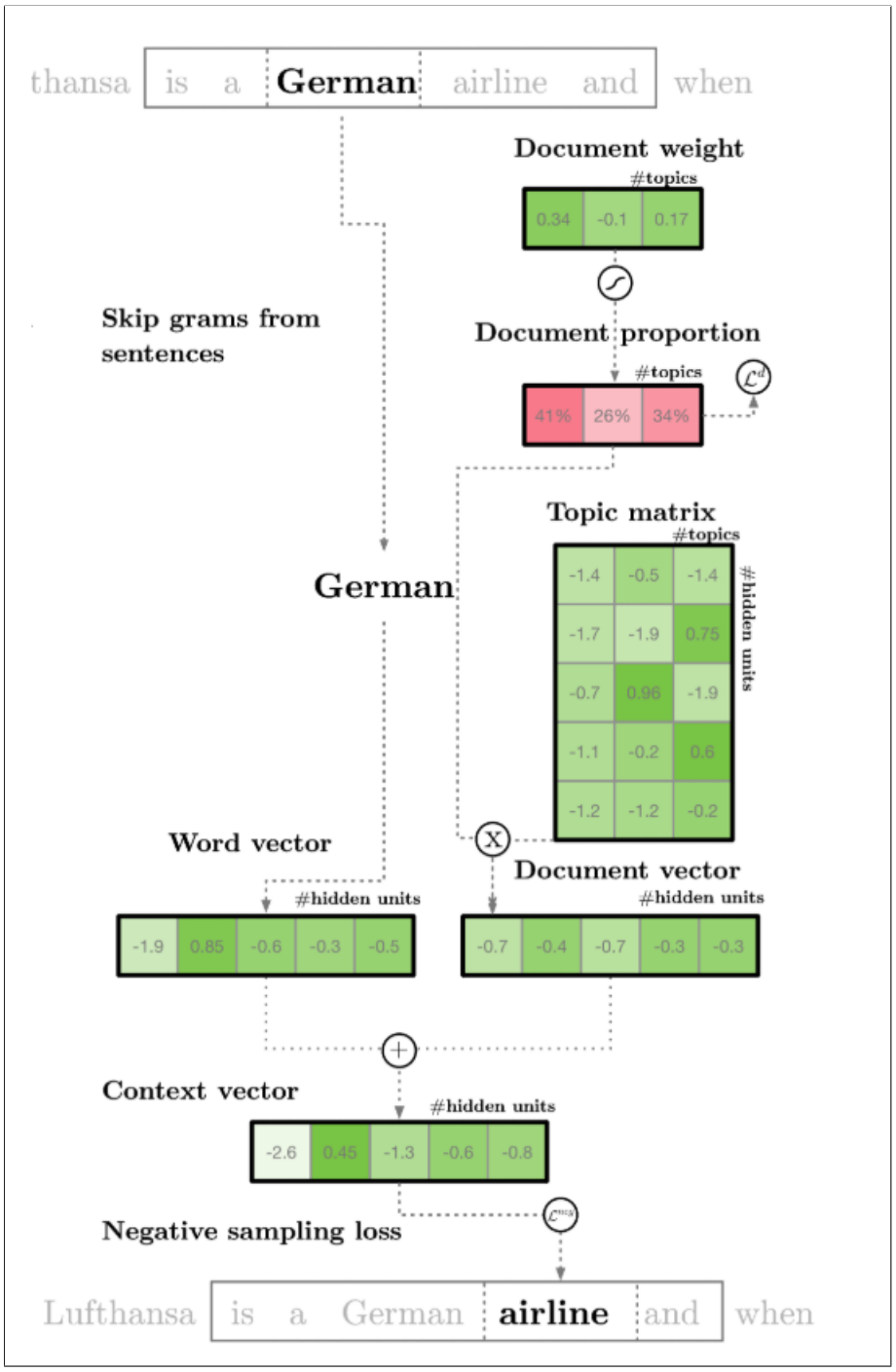

Figure 2.4 – lda2vec architecture overview taken from [34]

In the following subsections we will further describe the models that are most recent and relevant to our contribution.

2.2.1 Artificial neural networks concepts

In this subsection we present the core concepts of the artificial neural architectures that we use hereinafter.

Artificial neural networks (ANNs) are computing systems inspired by biological neural systems. The main purpose of ANNs is "learning" to perform a task (for instance, solve a classification, regression, or auto-encoding task).

The ANNs can be seen as a set of connected artificial neurons, arranged in layers. In general we distinguish the following three types of layers:

— input layer: is composed of a set of input neurons that are fed with the initial data (typically represented as a vector of features), that will be transferred to subsequent layers.
— hidden layers: these are the layers located in between the input and output layers. They form the part of ANNs where most of the processing happens as they discover and encode relationships between features (inputs). Each layer will create synthetic or meta features by combining, in general non-linearly the outputs of the precedent layer, and the complexity grows with the number of layers.
— output layer: it forms usually the last layer of a neural network and will produce the outputs [2].

We will note in the remaining of the document, the concatenation of two vectors as follow $concat(a, b) = [a, b]$

The simplest form of neuron in an ANNs is the Perceptron[35], the simplest neural structure that is characterized with a single artificial neuron. The output of a Perceptron[35] is nothing but the weighted dot product of its entries: $WX + b$, where $X = [x_1, x_2, x_3, ...]$ is the input vector (in the hidden layers the input is usually the output of the precedent layer), $W = [w_1, w_2, w_3, ...]$ is the weights vector that contains the so-called synaptic weights which are tuned during the learning process, and b is the bias that can also be learned.

The architecture of a perceptron neuron is illustrated in the figure 2.5.

2. An Artificial neural networks can have multiple inputs and outputs layers at different level of the network

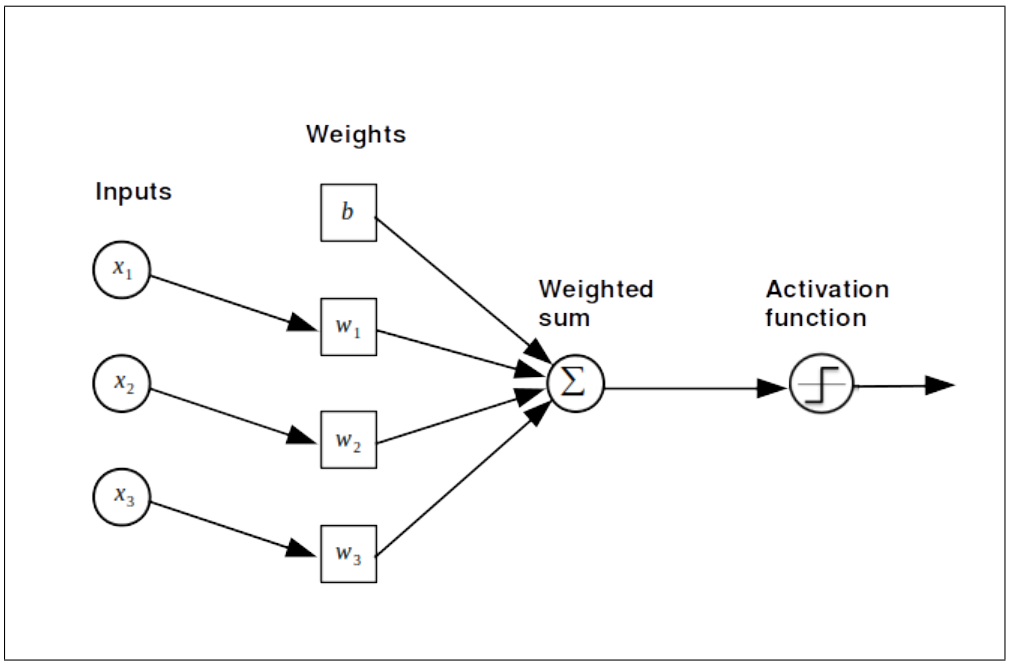

Figure 2.5 – The architecture of a Perceptron.

The Perceptron is a linear classifier characterized with a linear discriminant function. It means that the problem must be linearly separable for the Perceptron to work properly. In order to overcome this limit, more neurons or a non-linear activation function can be used [36] (we note that each neuron in an ANN has its own weight). The figure 2.6, shows the architecture of a Multi-Layer Perceptron. An activation function $f(x)$, is applied on the output of a neuron x. Among the most famous activation functions we can list the following:

— Sigmoid: $\frac{e^x}{e^x+1}$, the Sigmoid function restricts the output between 0 and 1, sigmoid function is usually used for binary classification.

— Hyperbolic Tangent: $\frac{2}{1+e^{-2x}} - 1$ the Hyperbolic Tangent function restricts the output between -1 and 1.

— Softmax: $f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$, where K is the size of x. The Softmax function restricts the output between 0 and 1 and can be seen as a probability distribution in case of multi-outputs ANNs, which means that the outputs sum up to 1. For example lets assume we have an ANN $f(x)$ with 3 outputs $y = [y_1, y_2, y_3]$ attached to softmax activation function (each output is a different class) $f(x) = [P(y_1|x), P(y_2|x), P(y_3|x)]$, where P is the probability to belong to a given class y_i. The *softmax* function works well with multi-output ANNs (Multi-class catego-

rization tasks). We note in the case of *softmax*, in the function $f(x)$, x is a layer output (multiple neurons).

In order to train an ANN, we need to update the weights of all the nodes (neurons) in the network according to a loss function. The method used for updating the weights of a neural network is called back-propagation, introduced in [37].

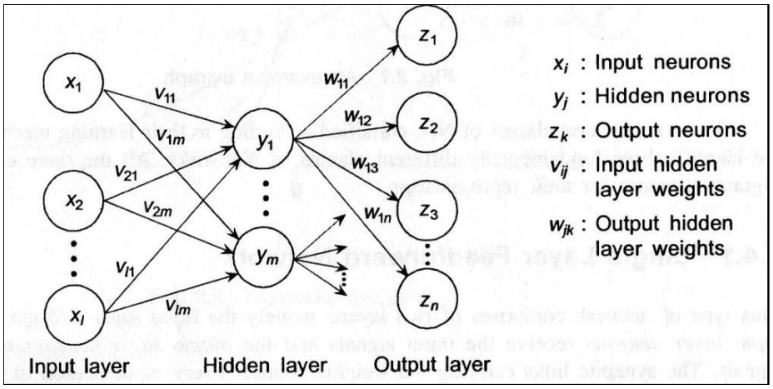

Figure 2.6 – Figure illustrating the architecture of a multi-layer perceptron taken from [38].

2.2.2 Word2vec

Word2vec [23] refers to a group of models that are used to produce word embeddings. In practice, they allow to produce word embedding from large corpora in an unsupervised fashion.

Word2vec embeddings are dense features vectors that capture the true semantics of words, to the extent that words with a similar meaning will have a similar representation: in other words, they will be close in the embedding space in which they will be projected. For example, "feeding" and "eating" are close (similar), while "eating" and "driving" are more distant.

The word-embeddings end up displaying very interesting relationships. For example, the result of subtracting or adding word vectors also carries a meaning. We can ask questions by using linear equations on simple words vectors : "King" - "Man" + "Woman" = X?, the result of such an equation would be the word "Queen", the closest word vector to the resulting vector X.
Word2vec architecture is close to an autoencoder[39] in the way that input and output

are the same words. Word2vec comes in two different architectures: CBOW (continuous bag of words [3]) and Skip-gram. CBOW creates the word embedding of a word by predicting its occurrence based on its context (a window of the words that occur before and after to it) as shown in the figure 2.7. The second architecture, Skip-gram, creates the word embedding of a word by predicting its context based on the word itself, as shown in figure 2.7. The distant words from the current one are given less weight by sampling them less in the training examples.

The prediction for the two architectures is done using a feed forward neural network [40] that contains three layers (input, hidden and output layers). The weight matrix of the hidden layer is used as words representation (word embedding).

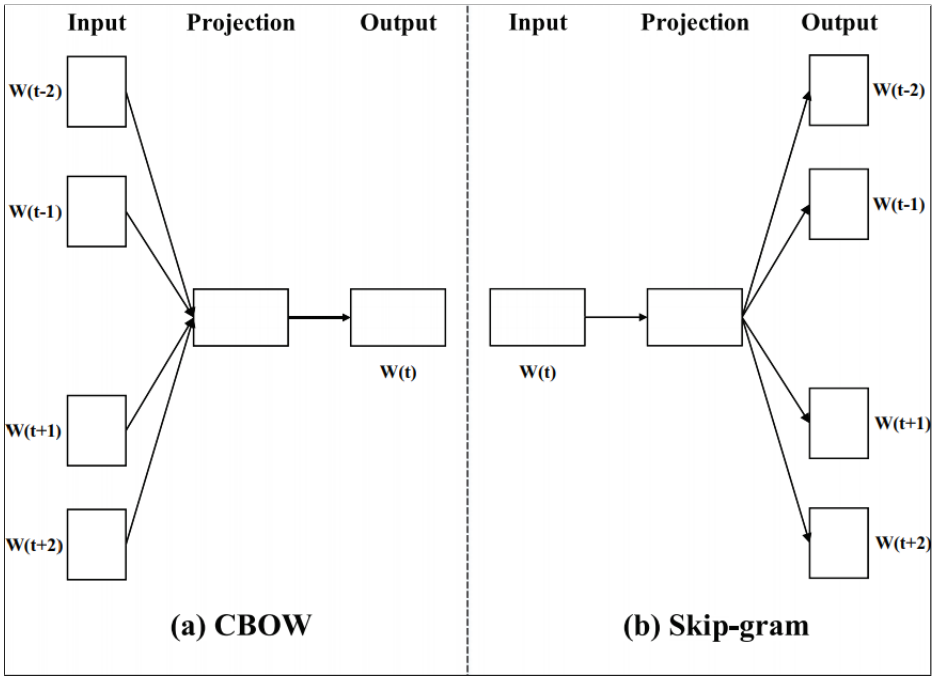

Figure 2.7 – Two word2vec architectures: CBOW and Skip-gram.

2.2.3 Latent semantic analysis

Latent semantic analysis (LSA) [22] is a natural language processing technique used to extract relationships between documents and terms they contain in order to find the "latent" concepts of the documents.

3. The notion of "bag of words" implies that the order of words is not taken into consideration

This is done by mapping both into what we can call a "concept" space (this technique relates to matrix factorization), which allows to produce document vectors in a reduced dimension.

LSA is done by applying an SVD (singular value decomposition) to a [terms x documents] matrix, and leads to a matrix factorization of the form:

$$A_{[m\times n]} = U_{[m\times r]} S_{[r\times r]} (V_{[n\times r]})^T \tag{2.3}$$

Where:

— A is the input matrix, in LSA A is a [terms x documents] matrix (n terms and m documents)
— U is the left singular vectors, an $[m \times r]$ (m documents, r concepts)
— S is the an $[r \times r]$ diagonal matrix that contains the singular values sorted in decreasing order (it can be seen as as the strength of each concept)
— V is the right singular vectors, an $[n \times r]$ matrix (n terms, r concepts)

NB: We note that the complexity of the SVD is $O(min(mn^2, m^2n))$

The following example will illustrate how LSA works. Let's assume that our "corpus" consists of the following "documents":

— $d1$ = "the white cat."
— $d2$ = "the dog is brown."
— $d3$ = "I love the dog."

We suppose that we use a term frequency vector as document representation, the result will be stored in matrix A figure 2.8.
The first step consists in decomposing the Matrix A into U, V^T and S (Eq. (2.3)). The result is shown in the figure 2.8.

The second step consists in applying a Rank k approximation (for this example $k = 2$), this is done by only keeping the k first columns U and V and the k first rows and columns of S associated to the k highest eigen values, as shown in figure 2.9.

The documents vectors in the reduced space are stored in the columns of the matrix V_k^T (the eigen vector values). In this example the coordinates of each document vector will be as follow:
$Vc1 = (-0.37, 0.93)$
$Vc2 = (-0.66, 0.26)$
$Vc3 = (-0.66, -0.26)$

$$
\begin{array}{l}
\textit{brown} \\
\textit{cat} \\
\textit{dog} \\
\textit{i} \\
\textit{is} \\
\textit{love} \\
\textit{the} \\
\textit{white}
\end{array}
A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}
\quad
U = \begin{bmatrix} -0.26 & -0.17 & 0.50 \\ -0.14 & 0.60 & 0.00 \\ -0.51 & -0.33 & -0.00 \\ -0.26 & -0.17 & -0.50 \\ -0.26 & -0.17 & 0.50 \\ -0.26 & -0.17 & -0.50 \\ -0.66 & 0.26 & 0.00 \\ -0.14 & 0.60 & 0.00 \end{bmatrix}
$$

$$
S = \begin{bmatrix} 2.56 & 0.00 & 0.00 \\ 0.00 & 1.56 & 0.00 \\ 0.00 & 0.00 & 1.41 \end{bmatrix}
\quad
V^T = \begin{bmatrix} -0.37 & -0.66 & -0.66 \\ 0.93 & -0.26 & -0.26 \\ -0.00 & 0.71 & -0.71 \end{bmatrix}
$$

Figure 2.8 – SVD decomposition of the A matrix: $A = USV^T$

$$
U \approx U_k = \begin{bmatrix} -0.26 & -0.17 \\ -0.14 & 0.60 \\ -0.51 & -0.33 \\ -0.26 & -0.17 \\ -0.26 & -0.17 \\ -0.26 & -0.17 \\ -0.66 & 0.26 \\ -0.14 & 0.60 \end{bmatrix}
\quad
V^T \approx V_k^T = \begin{bmatrix} -0.37 & -0.66 & -0.66 \\ 0.93 & -0.26 & -0.26 \end{bmatrix}
$$

$$
S \approx S_k = \begin{bmatrix} 2.56 & 0.00 \\ 0.00 & 1.56 \end{bmatrix}
$$

Figure 2.9 – The U_k, S_k, V_k^T matrices after applying a Rank $k = 2$ approximation.

In order to obtain a vector of coordinates Vq in the latent space for new documents q (documents that have not been incorporated into the initial [terms x documents] matrix A), we use the following equation:

$$Vq = q^T U_k S_k^{-1} \tag{2.4}$$

Where q is the query vector (the term frequency vector representing the new document). For our example, let's take the document $d4$ = "I love the cats.". Its term frequency vector is: $q = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$.
By applying the equation Eq. (2.4) to the vector q. The coordinates of the document vector in the latent space for the document q will be: $Vq = (-0.51, 0.33)$.

When working with big datasets we can use what we call "Truncated SVD". It is applied by only calculating the k column vectors of U and k row vectors of V^t corresponding to the k largest singular values. A faster way is to use randomized matrix approximation technique to perform a randomized SVD [41], which can be summarized as follow: First we calculate a matrix Q (Q is calculated using a collection of random vectors) with k columns (small number of columns) where $A \approx QQ^T A$ assuming such a Q exist.

Then, $B = Q^T A$ is constructed. Where B is a relatively small matrix (much smaller than A), a standard SVD is then applied to compute B. And thus $B = RSV^T$ (equation Eq.(2.3)) and as $A \approx QQ^T A$, we get $A = Q(RSV^T)$

2.2.4 Convolutional Neural Networks

Convolutional Neural Networks, CNNs for short, is a central model in many computer vision systems. The concept was introduced by Kunihiko Fukushima in [42] by implementing the mathematical formulation of Hubel and Wiesel's work on simple and complex cells in the human visual cortex. Later on it was used to introduce the first computational model for visual pattern recognition [10].

The CNNs were popularized by Yann LeCun et al. in [43] who experiment them in document recognition tasks. Following the development of word embeddings [24], [44] and its ability to represents word in dense fixed size vector while catching the semantics of words, CNNs start to be popular in the NLP community.

CNNs were originally designed to work on image matrices (2-dimensional arrays), but in this subsection we describe the one-dimensional CNNs used in NLP tasks. The CNNs can be seen as combination of two components/layers a convolutional layer and a pooling layer. The main objective of a convolutional layer is to extract local features from a sliding window on a sequential input (a sequence of word vectors for instance). This is done by mapping the input into a feature map in the form of a set of filters. Each filter will detect a pattern in the input by applying a convolution filter. The convolution layer can be seen as a convolution applied to a perceptron neural network with filters as weights.

The convolution layer formula is shown in the equation Eq.(2.5):

$$\begin{aligned} F_i^{sw} = g(\sum_{d=1}^{D_x}\sum_{w=1}^{D_w}(W_i \odot X_{(sw,sw+D_w)})), sw \in [1, T - D_w + 1] \\ F_i = [F_i^1, ..., F_i^{T-D_w+1}], F = [F_1, ..., F_N] \end{aligned} \tag{2.5}$$

Where W_i is the weight matrix of the i-th filter, N is number of filter, d is the input dimension index (also called channel), w is the window index; D_x is the input dimension (word embedding dimension in NLP), D_w is the sliding window size, $X_{(sw,sw+D_w)}$ is the sliding window (input sub-sequence) where sw is the sliding window index and T is the sequence length. The $\odot$ operator refers to the term by term multiplication. An activation function g is then applied on the output: rectified linear units (ReLUs) are recommended over the traditional activation functions for convolutional layers[45].

The equation Eq.(2.5) will be referred to as a function $f_{cnn}(W, X, sw)$ in the remaining part of this document.

$$ReLUs(X) = max(0, X) \tag{2.6}$$

From equation Eq.(2.5) we notice that the sliding window convolution layer steps through the input from left to right. Applying the filter iteratively causes the input borders to not be only exposed to the edge of the filter. We call this effect the "Border Effect Problem". In order to overcome this issue, padding is generally used. This is done by adding "padding vectors [4]" on each edge of the input, allowing the elements on the edge of the input to be exposed more than once on each filter.

4. A vector populated by zeros

The pooling layer is connected to the output of the convolutional layer and used to reduce the dimension of the of the convolved features by selecting the most important features. This also reduces the computational cost of the training and minimizes the risk of over-fitting. Pooling is done by slicing the input and applying the pooling function on each slice.

There are two main types of pooling: "Max Pooling" and "Average Pooling". The Average pooling performs down-sampling by dividing the input into regions and computing an average values for each region. While the max Pooling do the same but output the maximum values in each region.

Lets take as an example the sentence: "I am watching a movie", lets assume that each word is represented by a 2-dimensional vector. The figure 2.10 illustrate the use of a convolution layer with a size 2 sliding window (we don't use any activation function for this example)

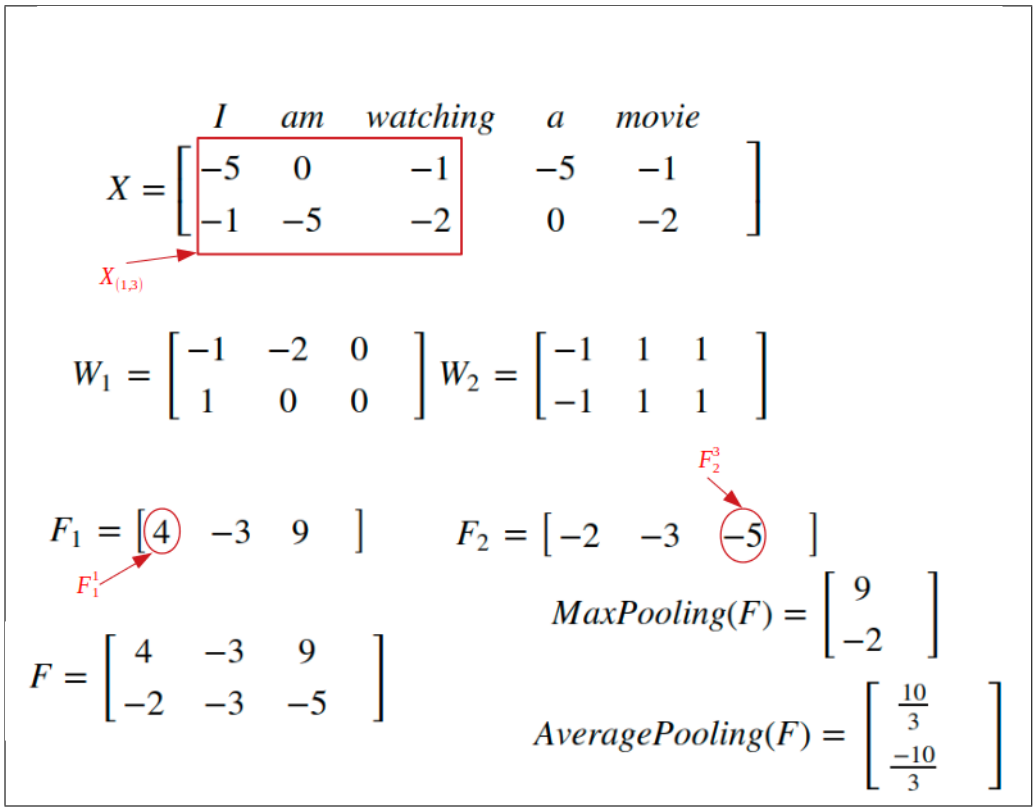

Figure 2.10 – CNN example that processes the sentence "I am watching a movie"

CNN model is usually built by stacking multiple pairs of convolutional and pooling layers, each subsequent layer will progressively aggregate simpler feature sets which

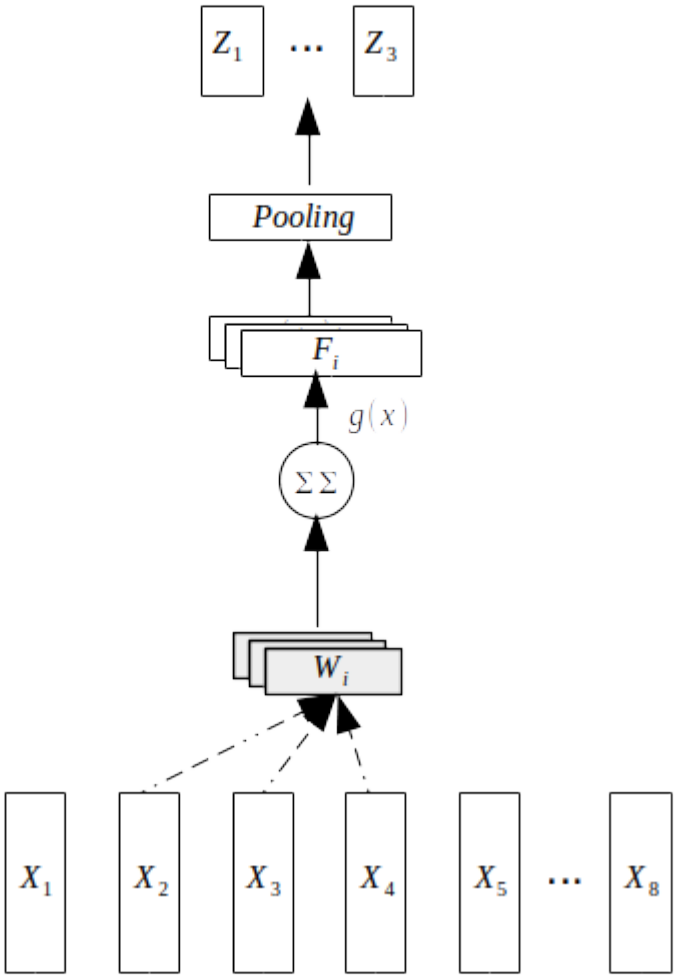

Figure 2.11 – Illustration of a CNN layer: in this example, the convolution has a window size of 3 and a pooling layer with 3 slices.

can be successfully used for classification.

2.2.5 Recurrent Neural Networks

In order to fully represent a document, we need to represent each word according to a model of its previous context (previous words). This mechanism can be referred to as persistence of "thought".

Traditional neural networks cannot perform such a mechanism (we note that CNNs can preserve local sequentiallity), and this has been considered as a major shortcoming. For example, if we want to put a label on each word in a sentence or address any other task that relies on sequence of terms, we need to manage specifically this sequentiality.

Recurrent neural networks address this issue. They integrate loops enabling the persistence of information.

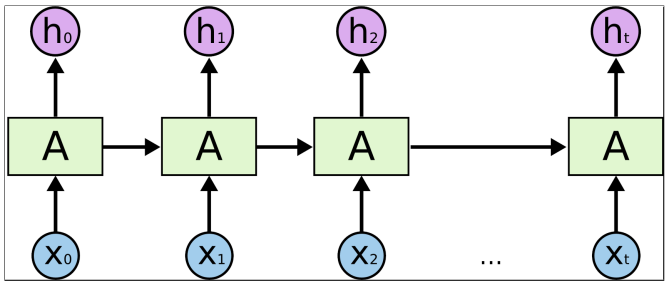

Figure 2.12 – Illustration of an unfolded RNN Layer

A loop allows information to be passed from one step of the network to the next, as shown in the diagram above (Fig.2.12). Recurrent neural networks are similar to traditional neural networks: they can be described as a chain of multiple copies of the same network architecture, each copy passing information to its successor, in a chain like architecture enabling to work with sequential data. They have been applied successfully to several problems such as speech recognition, language modelling, language translation, image captioning, etc.

This kind of recurrent architecture works well if the objective is only to consider recent information in order to perform the task. In other words, it works well if the interval between the relevant information and the objective is small, meaning that this architecture is associated to a short memory capability. For example, consider the task of predicting the next word based on the previous context. If we are trying to predict

the word "bananas" in the sentence "the monkey eats bananas", we do not need any longer context. In some situations however, we may need more context. For example when we try to predict the word "gym" in the sentence "In order to stay in shape, I go every day to the gym.". In this case, close context suggests that the next word is probably a place and in order to have some information about the right word, we need to know the context of sport (shape), from the start of the sentence.

The figure 2.13 an example of long context.

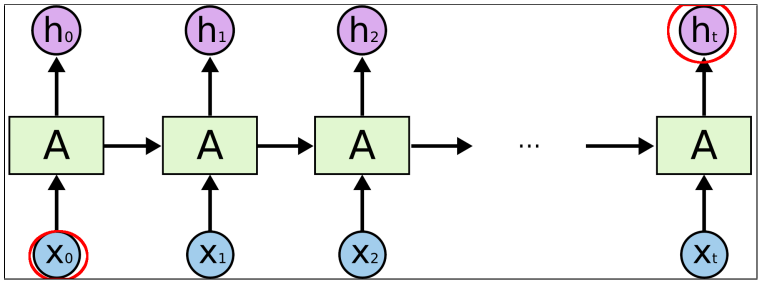

Figure 2.13 – Figure illustrating an example of long context (long-term dependencies) in RNNs.

Unfortunately, as the interval grows between the relevant information in the sequence and the word to predict, RNNs performance degrades as it becomes harder to connect the relevant information to the objective word. This problem was specifically explored in [46].

2.2.6 Long Short Term Memory networks

Long Short Term Memory networks (LSTMs)[47], are special kind of RNNs capable of learning long-term dependencies, thus it addresses the problem cited above while storing relevant information for long periods of time.

LSTMs have a similar architecture to that of RNNs. However they implement a recurrent component that is more complex as it contains four neural networks layers as shown in in figure 2.14. Each rectangle highlighted in this figure is a Neural Network layer (Perceptron), $tanh$ refers to a Hyperbolic Tangent activation function and σ stands for a Sigmoid activation function.

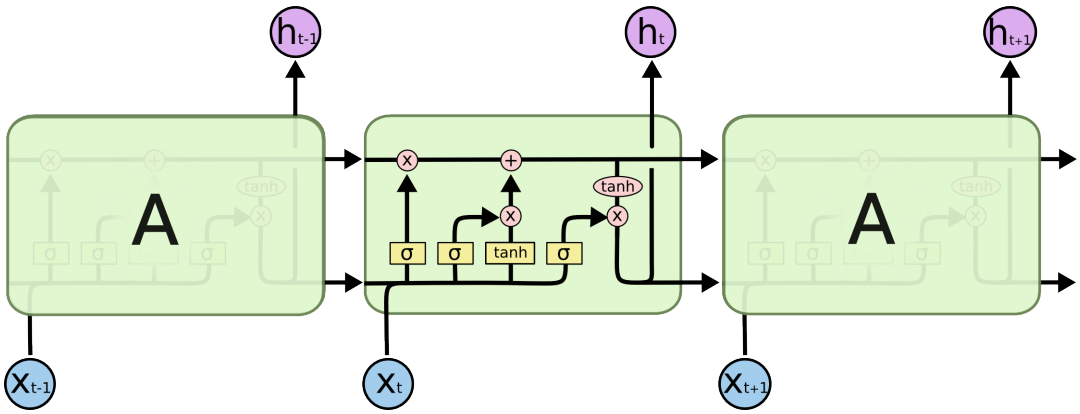

Figure 2.14 – Illustration of an unfolded LSTM Layer

The core idea behind LSTMs is the use of cell states and internal gates mechanisms implemented by sigmoids, that monitor the flow of information. By training the network, the gates learn which information to keep and which to discard. They are composed with a sigmoid layer and a point-wise multiplication operation. Keeping in mind that a sigmoid function will output numbers between zero and one, such gating architecture controls how much information is passing through. An LSTM has three of such gates.

We can look at a LSTM as a three steps process, as shown in the figure 2.15:

1- The first step of an LSTM is the "forget gate layer" (sigmoid layer), it decides whether it keeps or throws away the information from the memory cell state.

The forget gate formula is given bellow:

$$f_t = \sigma(W^{hf} h_{t-1} + W^{xf} x_t] + b_f)) \tag{2.7}$$

Where W^{xf} and W^{hf} are the weights matrices, b_f is the bias, h_{t-1} is the hidden state vector and x_t is the input, the output of f_t will be in the $[0, 1]$ interval.

2- The second step allows for deciding which information to store in the cell state; it is called "input gate". And is composed of two parallel layers: a "sigmoid layer" that monitors what information needs to be added to the cell state based on h_{t-1} and x_t as described in this formula:

$$i_t = \sigma(W^{hi} h_{t-1} + W^{xi} x_t] + b_i) \tag{2.8}$$

the second layer a "tanh layer" is used to create a vector $\tilde{C}_t$ that contains new values

that could be added to the next step, as described in the following:

$$\tilde{C_t} = tanh(W^{hc}h_{t-1} + W^{xc}x_t] + b_c) \tag{2.9}$$

The two parallel parts are combined using a point-wise multiplication, the result is used as an update to the state by combining it to C_t via an addition operation, as illustrated in this formula:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C_t} \tag{2.10}$$

Where $\odot$ is a point-wise multiplication.

3- The last step corresponds to the "output gate". In this step the output h_t is evaluated based on a filtered version of the cell state C_t. So a $tanh$ function is applied on C_t the result will then be combined to the "output gate" o_t using a pointwise multiplication, where o_t and h_t are defined as follow:

$$o_t = \Sigma(W^{ho}h_{t-1} + W^{xo}x_t] + b_o) \tag{2.11}$$

$$h_t = o_t \odot \tanh(C_t) \tag{2.12}$$

The figure 2.15 illustrate the combinaion of the three steps.

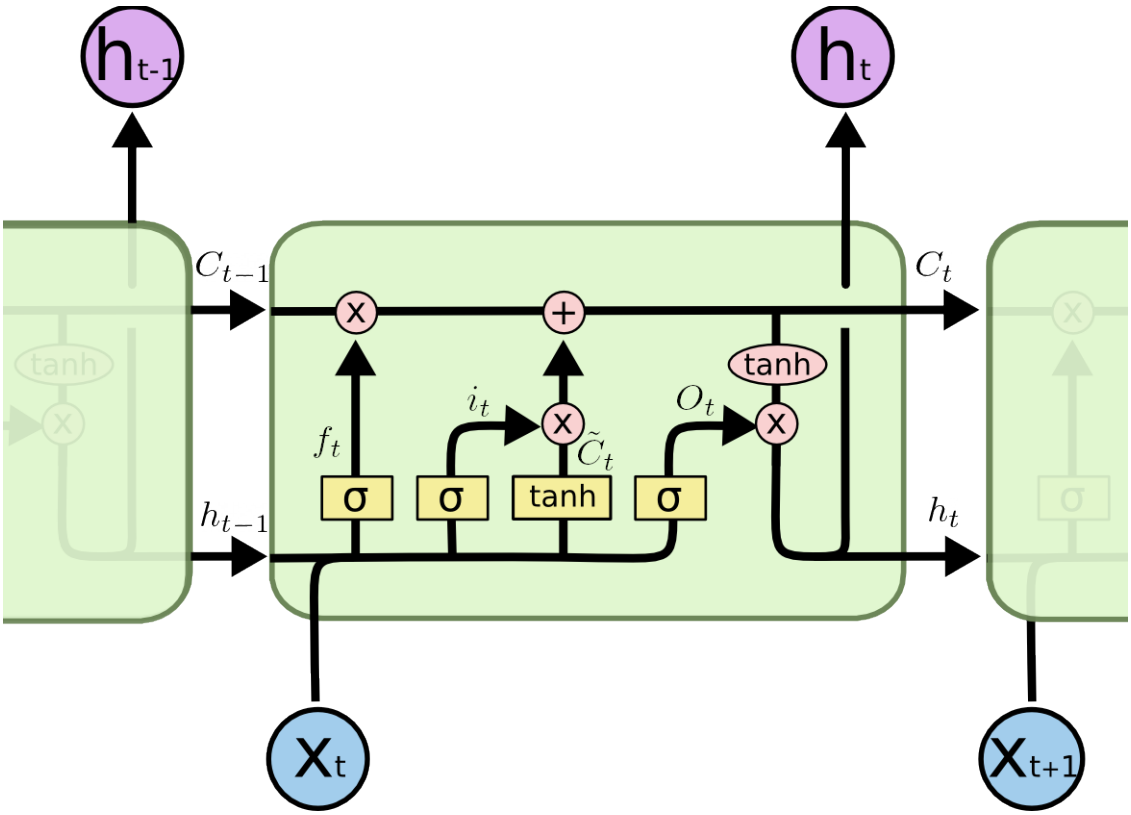

Figure 2.15 – Detailed illustration of an LSTM cell.

2.2.7 Attention mechanism

When the problem involves long-term dependencies (in natural language processing in particular), RNNs (even when considering LSTMs/GRUs [9]) suffers from a number of difficulties:

— Information vanishing problem, caused by the gate mechanism while more information is added to the single hidden state vector available [5].

— The fact that RNNs have to process the elements of a sequence one by one, therefore this process cannot be parallelized[48].

Attention mechanism was developed to address those problems.

The motivation behind attention mechanism is to model the way we pay visual attention to different regions of an image or correlated words into a sentence. For example while reading the sentence: "I am watching an action movie" we put, in general, more attention on the pair of words (watching, movie) than on the pair (watching, action):

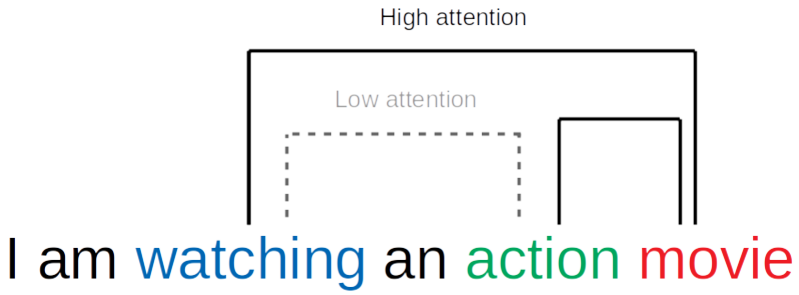

Figure 2.16 – One word “perceived” differently when associated to different words in a same sentence.

Relationships (semantic affinities) between words is the cause for such perception gap, as when we encounter the word "watching" we expect to encounter a scene or movie related word very close after the word "watching". The word "action" is more related to the word "movie" and not much with the world "watching".

Attention mechanism aims to mimic something we humans do while reading a text, as we tend to focus more on certain words that we believe are more relevant and contribute more to the subject of the text.

In other words attention mechanism allows a neural network to focus on relevant parts of the input. Attention can be seen as a vector of importance weights that characterize elements (lets say words in a sentence) in relation to other elements in a sequence (how strongly it is correlated with or “attends to” as expressed in [48]).

Attention mechanism was developed to improve the performance of the Encoder-Decoder RNNs on machine translation tasks, presented by Dzmitry Bahdanau, et al. in [5] as a solution to information vanishing due to the fact that an encoder will try to compress all the necessary information from source sentence into a fixed-length vector (the longer the sentence the more information is lost during the compression).

Instead of encoding the input sequence into a single fixed context vector that will be passed to the decoder which cause a bottleneck as shown in figure 2.17, the attention

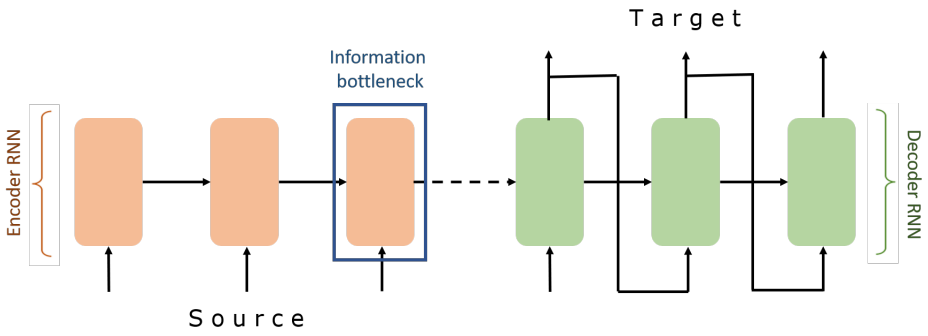

Figure 2.17 – Illustrating the information bottleneck in an auto-encoder architecture.

model develops a context vector that is filtered specifically for each output time step. In addition, the decoder will receive all the context vector (RNN's outputs).

The use of attention in NLP was popularized by Dzmitry Bahdanau et al. [5] on English-to-French translation tasks. The proposed architecture is shown in figure 2.18

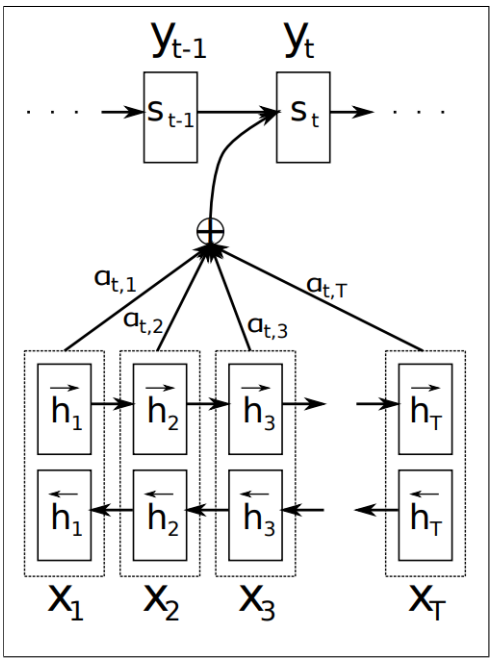

Figure 2.18 – Example of the Attention mechanism taken from [5].

In a classical RNN the conditional probabilities are computed as follows given the context vector c (in RNNs the context vector is the sentence/document embedding which is the last hidden state h_t of the encoder RNN) and all the previously predicted

words $y_1, ..., y_{t-1}$:

$$p(y_t|y_1, ..., y_{t-1}, c) = g(y_{t-1}, s_t, c) \tag{2.13}$$

where g is a non linear function (the output of a multi-layer in most cases) that outputs the probability of y_t and s_t is the hidden state of the RNN.

However in this model architecture that corresponds to the so-called "Vanilla attention", each conditional probability is defined as follow:

$$p(y_i|y_1, ..., y_{i-1}, x) = g(y_{i-1}, s_i, c_i) \tag{2.14}$$

Where s_i is an RNN hidden state for the step i, computed as follow:

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \tag{2.15}$$

We note that each probability is conditioned on specific (different) context vector c_i for each output word (target word) y_i.

The context vector c_i depends on the annotation (outputs) of the RNN[5], each annotation h_i is an output of the bidirectional RNN that contains information about the whole sentence (input sequence) with a strong focus on the i-th word and the words surrounding it.

Then, the context vector c_i is computed as a weighted sum of these annotations h_i and the attention weight a_{ij}:

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_i \tag{2.16}$$

The attention weights a_{ij} are computed as follow:

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{2.17}$$

Where :

$$e_{ij} = \alpha(s_{i-1}, h_j) \tag{2.18}$$

We can note that the equation Eq.(2.17) is a softmax activation function.

5. in the paper describing the vailla attention a GRU network was used ($h_1, ..., h_{T_x}$)

The alignment model α scores how well the inputs around the j-th word and the word at the i-th position match. The alignment is based on the hidden state s_{i-1} of the decoder RNN and the annotation h_j of the encoder RNN.

The model α is implemented as a feedforward neural network which is jointly trained with all the other components of the proposed system.

Transformer

Transformer architecture was introduced by Vaswani and al [48] in the paper "Attention is all you need". As the title suggests the main focus of this model is to remove the traditional RNN encoder-decoder in traditional architecture for machine translation that are particularly costly to train. We will only describe the encoder part since it is the part that performs the vectorization task, the focus of this chapter.

The overall architecture of a transformer is shown in figure 2.19.

The Transformer model uses a particular form of attention mechanism called the “Scaled Dot-Product Attention” which is defined as follow:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.19}$$

$$\begin{aligned} Q &= XW^Q \\ K &= XW^K \\ V &= XW^V \end{aligned} \tag{2.20}$$

Where $Attention$ function can be seen as mapping a query (Q), and a set of key(K)-value(V).

The output of the $Attention$ function takes the form of a weighted sum of quantities that depend on the query and the corresponding keys. Therefore QK^T is the inner product (homogeneous to a similarity measure) between the projections of the word vectors X into W^Q and W^K spaces: this evaluation can be interpreted as calculating the correlation between a given word and all the others. Furthermore, applying a projection of X into W^V allows to have more control on word level representation by tuning (increasing/decreasing) its dimension.

The vectors K, Q are in d_k dimensions and V is in d_v dimensions.

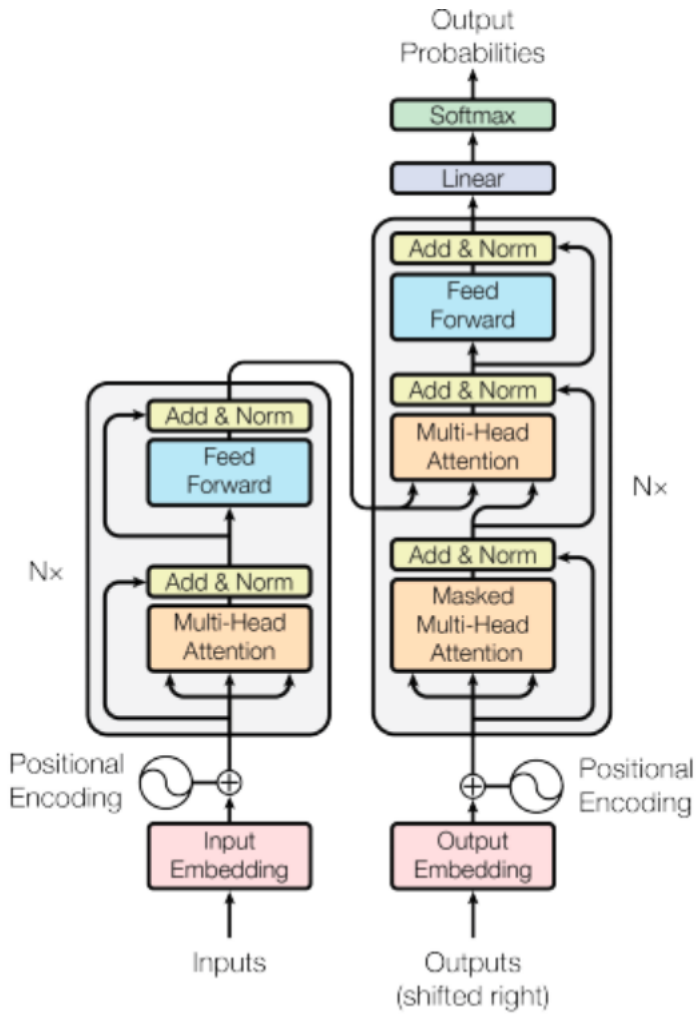

Figure 2.19 – Transformer model architecture taken from [48].

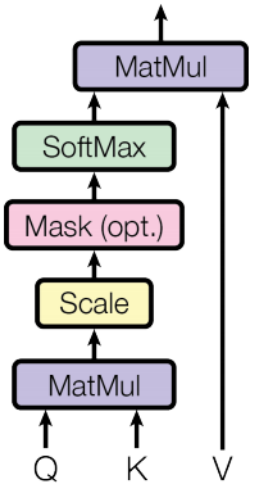

Figure 2.20 – Illustration of scaled dot product taken from "Attention is all you need" [48].

The difference between the scaled dot-product attention (as shown in figure 2.20) and Self-attention (intra-attention) [49] is the scaling factor $\frac{1}{\sqrt{d_k}}$. According to [48] for large values of d_k the scaling factor seems to improve the performance.

Instead of using a single attention function, the transformer architecture uses a Multi-head attention. This is achieved by linearly projecting (K, Q, V) n times (where n is the number of heads) with different, learned linear projections. In practice this is done by initializing n different matrices (W_i^K, W_i^K, W_i^V) for projecting (K, Q, V), in each version of (K_n, Q_n, V_n) the attention is calculated in parallel, which allows the attention layer to exploit information from different positions and different representation sub-spaces. The outputs are then concatenated and weighted as shown in figure 2.21 and Eq.(2.21)

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_n)W^O = Z \tag{2.21}$$

Where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$

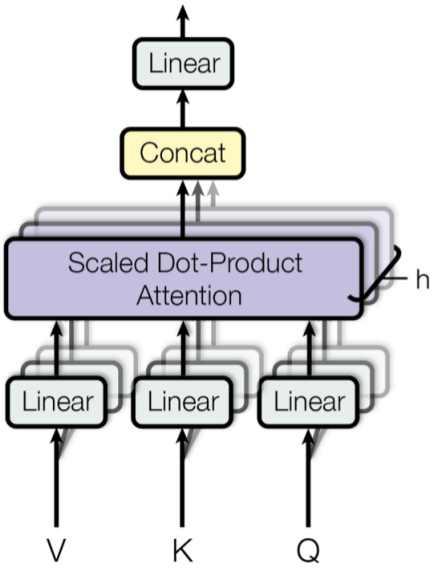

Figure 2.21 – Transformer model architecture taken from "Attention is all you need" [48]

Since the Transformer model contains no recurrence nor convolution, in order to keep information about the order in the sequence a positional encoding is applied. The positional encoding vectors have the same dimension than the word embedding and are calculated as follow:

$$\begin{aligned} PE_{pos,2i} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{pos,2i+1} &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{2.22}$$

Where pos is the position, i is the dimension and d_{model} is the word embedding dimension. The resulting vector is then added to the word vector.

$$x_{pos} = x^e_{pos} + t_{pos} \tag{2.23}$$

Where x_{pos} is the positional weight, after applying the positional encoding, x^e_{pos} is the word embedding vector and t_{pos} is the positional embedding vector which corresponds to the pos-th position.

The output of the multi-head attention sub layer $Z = (z_1...z_n)$ are then added to the word annotation $X = (x_1, ..., x_n)$. The resulting vector is then normalized [50].

The sentence vector can be obtained by combining the the vectors $[z_i, ..., z_n]$ using a pooling layer that can takes the form of a feed forward neural network [6].

2.2.8 Hierarchical attention

Hierarchical Attention Network (HAN) [11] has been designed for text classification tasks. The attention mechanism that this model implements is close to the vanilla attention concept (introduced in section 2.2.7), with the difference that lies in the use of a context vector.

The HAN model considers a document as a sequence of sentences and a sentence as a sequence of tokens (usually words). Hence the name "Hierarchical Attention". According to this model the attention is calculated at two levels (at word and sentence levels).

At word level the HAN model uses a bidirectional GRU network word encoder ([5]). The word encoder assigns word w_{ij} (i is sentence identifier and j is the word identifier) to vector x_{ij}. This word encoder consists of an embedding matrix W^e, $X_{ij} = W^e_{w_{ij}}$, where w_{ij} is the word entry in the dictionary [9], [51]. W^e is trained along side with the model. The word annotations h_{ij} benefit from both directions of the "bidirectional GRU" as explained in section 2.2.7. The GRUs are labeled as: $\overrightarrow{GRU}$ for the unit that reads a sentence from w_{i1} to w_{iT} and $\overleftarrow{GRU}$ for the one that reads a sentence from w_{iT} to w_{i1}. Here T is the length of a sentence (the number of words).

The annotation h_{ij} of the word w_{ij} is obtained by concatenating the forward hidden state $\overrightarrow{h_{ij}}$ and backward hidden state $\overleftarrow{h_{ij}}$ ($h_{ij} = [\overrightarrow{h_{ij}}, \overleftarrow{h_{ij}}]$).

Where:

$$\begin{aligned} \overrightarrow{h_{ij}} &= \overrightarrow{GRU}(X_{ij}) \\ \overleftarrow{h_{ij}} &= \overleftarrow{GRU}(X_{ij}) \end{aligned} \tag{2.24}$$

The word attention is then calculated as follow:

$$u_{ij} = tanh(W_w h_{ij}) \tag{2.25}$$

$$a_{ij} = \frac{exp(u_{ij}^T u_w)}{\sum_j exp(u_{ij}^T u_w)} \tag{2.26}$$

Where u_w is the context vector that can be seen as a high level representation of a fixed query expressed as: "what is the informative word?" [52], [53] (in the context of our classification problem).

u_w is randomly initialized and trained along side with the rest of the model.

The sentence vector is then calculated by summing words annotation h_{ij} weighted by the attention a_{ij}, as shown in equation Eq.(2.27)

$$s_i = \sum_j a_{ij} h_{ij} \tag{2.27}$$

At sentence level, the encoder works basically the same way as for the word encoder, taking s_i as the bidirectional GRU input. The sentence encoder architecture can be summarized by the following equations:

$$u_i = tanh(W_s h_i) \tag{2.28}$$

$$a_{ij} = \frac{exp(u_i^T u_s)}{\sum_i^e xp(u_i^T u_s)} \tag{2.29}$$

$$v = \sum_i a_i h_i \tag{2.30}$$

Where h_i is the sentence level bidirectional GRU output and v is the document vector (in equation Eq.(2.30)).

The classification is then achieved using a feed forward neural network.

The overall architecture of the Hierarchical Attention Network (HAN) is illustrated in the figure 2.22.

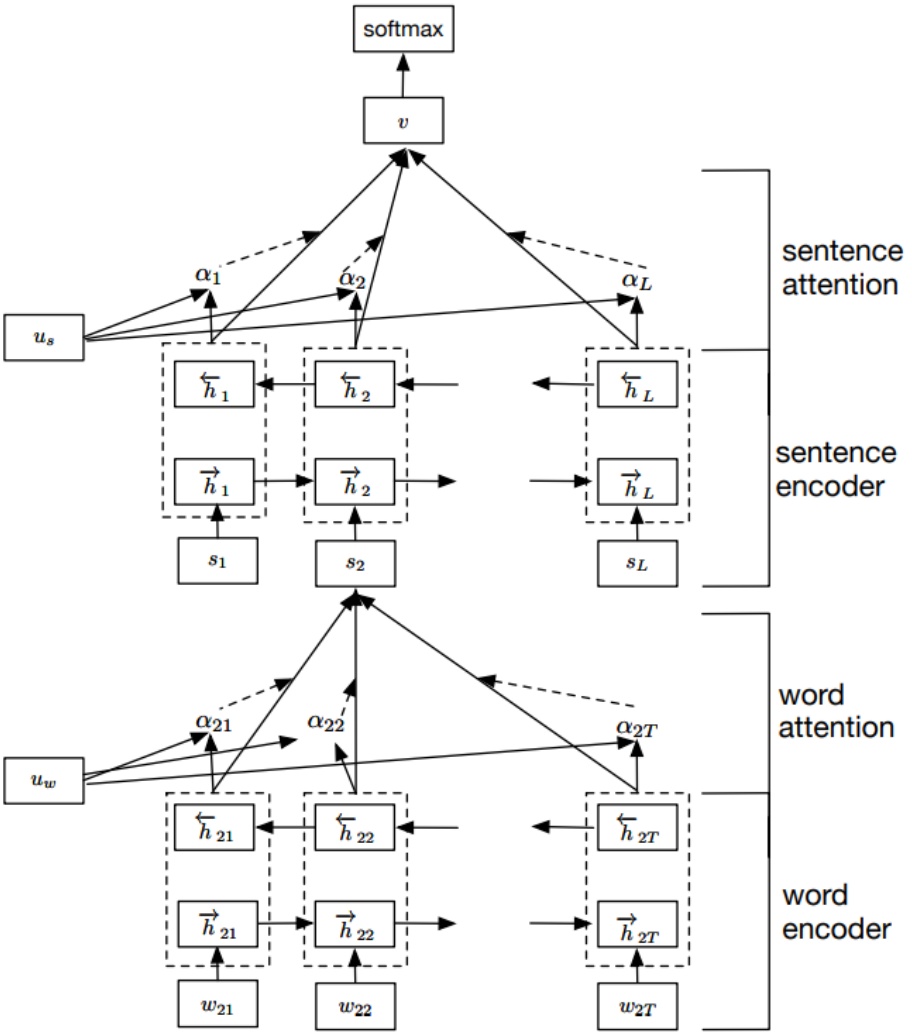

Figure 2.22 – Hierarchical Attention Networks architecture taken from [11].

2.3 Contribution

Document vectorization is a critical step for multiple tasks at Octopusmind company (recommender system design, clustering document and classification tasks, etc.).

The Vectorization model must meet certain criteria before it can be integrated into a production process, in particular:

— The model has to be light weight given the limitation of the resources, in particular the storage capacity and the processing hardware the company can afford.

— The vectorization process has to be efficient, so it can be used in a real time advanced query processing framework.

— The document embedding should be able to store semantic information, in other words the distance between two documents should reflect their semantic distance.

— The model must be compatible with the nature of the call for tender documents, for which most of the content is irrelevant to the filtering/retrieval tasks (we consider such content as noise).

— The sequentiality of words in a sentence is only important in a local context (a sliding window containing few words).

Our two following contributions address a vectorization at document level. Keeping in mind that the models had to answer the criteria above. The first one proposes to benefit from the combination of word2vec and LSA methods to construct a vectorization at document level that exploit two complementary views on the context of word occurrences. The model was light weight, relatively fast and stored very well semantic information but had issues handling noisy documents. The second contribution relies on a new model to capture attention, the so-called CNN-based attention mechanism, that is exploited hierarchically in order to construct a vectorization at document level from a vectorization defined at sentence level. This model has resolved the issues when working with noisy data.

2.3.1 LSA+W2V

Our first model referred, to as LSA+W2V [8], is constructed from the word2vec and LSA vectorizations that are defined at word levels. This model aims at taking advantage of these two complementary views on lexical semantics. Assuming that the general meaning of a text is the combination of the words that compose it, the vectorial representation of a document (document embedding) can be constructed as a combination of the vectorial representations of the words that form the document. The naivest way to produce such a vectorial representation at document level, is to average the word vectors. The figure 2.23 illustrates an example with the sentence "Télécomunication optique et vidéo" ("optical telecommunication and video").

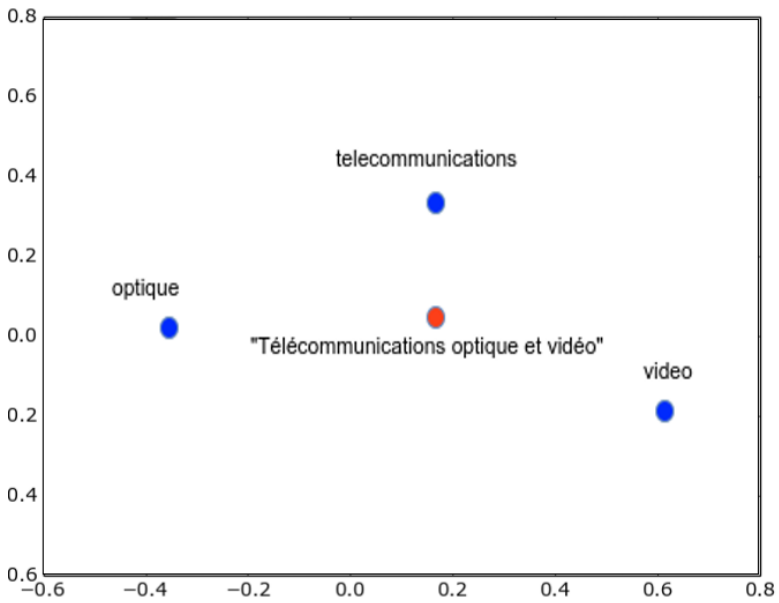

Figure 2.23 – Figure illustrating an example of word vector averaging result with the sentence "Télécomunication optique et vidéo"

The resulting mean vector can be seen as as a vector describing the general idea developed in the document. However, an average based aggregation comes with few disadvantages: as it tends to be sensitive to outliers (extreme values), the result could be irrelevant in the case of asymmetric data. Worse, in some cases the calculated average could fall in an area of the embedding space that is void, i.e. far from any observed word vectors: for example the mean value of -10 and 10 is 0 which is far from the two initial values and could generally correspond to a void area of the embedding

space.

Moreover, from the word2vec averaging representation of a document, it is difficult, if not impossible, to go back to the words contained in the text, even if we address this problem probabilistically. In the other hand, an LSA vector retains information about the most important words that characterise the texts in the context of a set (corpus) of documents.

It is the complementary of these two ways to address the context of word occurrences (locally around the word for word2vec, and globally in the context of a set of documents for LSA) that motivates our approach. By concatenating the word2vec mean and the embeddings provided by LSA, we obtain a low-dimensional vector representation at document level (document embeddings) that catches both the general semantics of a document (general meaning) and maintaining a lexical-conceptual description of this document. To our knowledge this combination has not been explored yet. The combination of word2vec and LSA is illustrated in the figure 2.24.

We made the choice of applying LSA on [documents x terms] matrix, weighted using *tf-idf* heuristic.

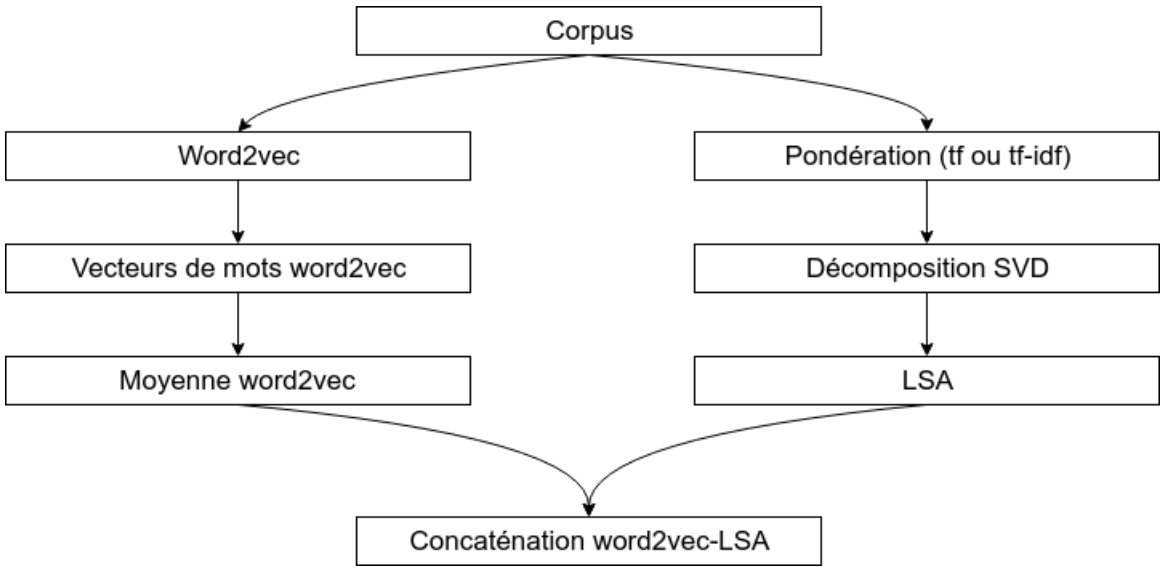

Figure 2.24 – Illustration of the way the combination of word2vec and LSA is achieved.

If we compare LSA+W2V to doc2vec which is basically a modified version of word2vec, it builds the document vector based on local context, while our model combines the benefits of using word2vec that is also based on local context words to LSA that provide a more global context, by keeping information about the words that compose the

document.

And compared to the combination of word2vec and LSA proposed in [31], our model is more memory efficient, therefore more practical while working with big corpora.

2.3.2 CnHAtt

The W2V+LSA model is time-consuming to build, as it depends on building two separate models (LSA and word2vec) that have respectively a time complexity of $O(Nlog(V))$ and $O(VNmin(N,V))$ where V is the number of unique words in the vocabulary and N is the number of document. But is able to generate lightweight vectors that somehow catch the document semantic. However it shows some limitation: in particular it assigns the same weight to all the words while creating the document vector, without considering their relevance to the considered task (even words that do not contribute to the main topic addressed by a document are still considered into the vector). In order to overcome those limitations, we propose a model that takes into account the hierarchical nature of a document (a document can be seen as a list sentences and a sentence as a list of terms) and also weighting the words and sentences according to their relevance to the topic of the document or to the task that is tackled (document classification for instance).

Our model CnHAtt (Convolution based Hierarchical Attention) can be compared to the HAT model [11], as it also has a hierarchical attention that applies the attention on two level, on a word level which allows to give each word a weight according to its local context and on a sentence level by giving each sentence a weight according to their relevance to classification task. But also takes advantage of the transformers[48] architecture by getting rid of the RNNs, which allows it to be easily parallelisable, therefore decreasing significantly the training and prediction time.

Instead of using self-attention or vanilla attention [5], we propose to use a CNN architecture to mimic the attention mechanism in a local context. The choice of a local attention mechanism is motivated by the fact that unlike global attention, it does not need to attend to all words, which is expensive, and slowdown the performance while working with long sequences [54].

To our knowledge, the CNN-based attention detailed herein after is novel and the concept of attention has not yet been addressed in such a manner.

In our DNN architecture, the multiple filters in CNN are acting like a multi-head

attention: each filter of the CNN deals with a specific representation that focuses on different aspects of the input.

In other words the CNN filters will focus on different patterns extracted from a subsequence of words, thus performing a local attention [54] instead of a global attention as implemented into the HAT model [11], in vanilla attention [5] or in the transformer architecture [48]. As we do not use any RNN layer, the attention A in our model is only conditioned by the input vector X.

Our model is dedicated to the construction of a document-level representation. We assume that a document is composed with S sentences and each sentence is composed with T words (including padding if necessary). w_{ij} represents the j-th word in the i-th sentence. We use padding in our model to ensure that each word vector in entry can be located at a center of a filter.

Our model first maps word w_{ij} to the embedding vector X_{ij} through a word encoder which consists of an embedding matrix W^e, $X_{ij} = W^e_{w_{ij}}$, where w_{ij} is the word entry in the dictionary [9], [51]. W_e is trained along side the model.

The choice between the *max pooling* and *average pooling* can be explained intuitively as follow:

— Global max pooling will give attention to words that are relevant to at least one CNN filter.

— Global Average pooling will give attention to words that are in general relevant to most CNN filters.

$$H_{ij} = f_{cnn}(W_t, [X_{ij-(sw-1)/2}, ..., X_{ij+(sw-1)/2}], sw) \tag{2.31}$$

The attention at a word level in our model is calculated according to Eq.(2.32):

$$a_{ij} = g([H_{i1}, ..., H_{iT}]) \tag{2.32}$$

Where f_{cnn} is a CNN layer, g is a function that selects which filters to consider (pooling layer), in our model we use a Global Average Pooling or a Global Max Pooling (we can also use a feed forward layer as a function). H_{ij} is the output of the convolution layer centered on the word w_{ij} (i is sentence id and j is the word id) .
Finally, n is the size of the window used by the word level CNN and F^w_k is k-th filter of the word level CNN,where F^w is word level filters matrix.

Then each dimension of the word embedding vector X_{ij} is multiplied by the corresponding attention weight a_{ij} the resulting vector Z_ij is then summed up to get the

sentence context vector C_i as illustrated in 2.25.

$$Z_{ij} = X_{ij} \times a_{ij} \tag{2.33}$$

$$C_i = \sum_{j=1}^{T} Z_{ij} \tag{2.34}$$

We note that the same convolution layer is applied to all the sentences.

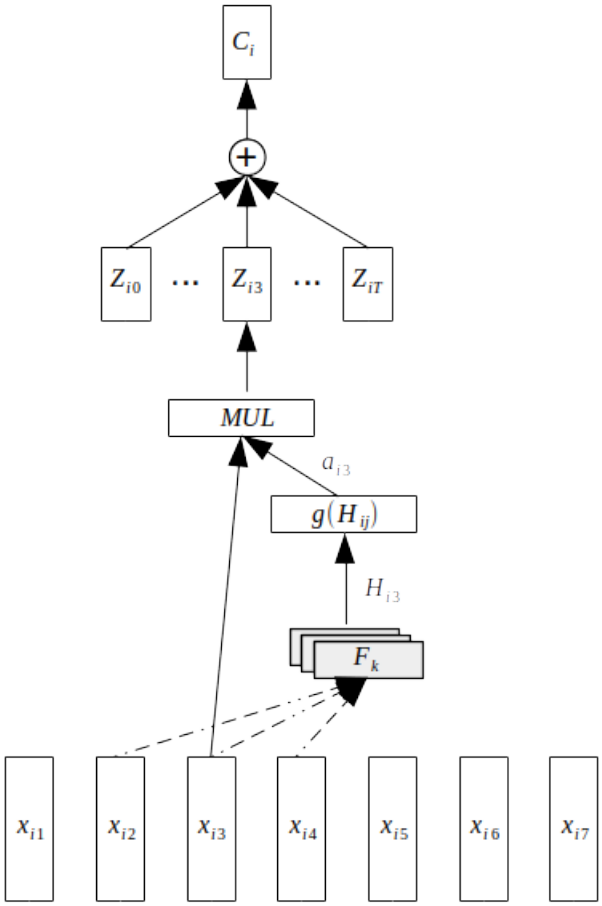

Figure 2.25 – Illustration of the convolution-based attention mechanism developed in our model at sentence level.

At the sentence level the model calculate the attention a_i according to the following

equations:

$$H_i = f_{cnn}(W_s, [C_{i-(sw-1)/2}, ..., C_{i+(sw-1)/2}], sw) \tag{2.35}$$

$$a_i = g([H_1, ..., H_S]) \tag{2.36}$$

where f_s if CNN layer, m is the size of the window used by the sentence level CNN and F_k^s is k-th filter of the sentence level CNN.
The representation vector of the document C is calculated the same way as the context vector C_i (sentence representation) as illustrated in figure 2.26.

$$Z_i = C_i \times a_i \tag{2.37}$$

$$C = \sum_{i=1}^{S} Z_i \tag{2.38}$$

To use our model in a classification framework, the document vector C is used as an input of any neural network architecture in charge of the classification task. In our case, we use a non linear MLP that is trained on the classification task along side the attention vectorization layers, in order to fine tune the word embedding.

We have also studied different variations of the proposed model: The first variation (CnHTr) is based on self-attention on a local context, this is done by creating a Key K, Query Q and a Value V but instead of using a dot product like in Eq.(2.20), we use three convolution layers ($f^Q(x), f^K(x), f^V(x)$) to create Q, K, V, where $f^Q(x)$ and $f^K(x)$ have the same sliding window size. This allows to calculate an attention that is based jointly on local and global context. It is also possible to mimic the multi-head model by using multiple ($f^Q(x), f^K(x), f^V(x)$) and concatenating the result of the attentions machanisms on each head, in the same way as in Eq.(2.21).

The second version is based on a rewriting of equation Eq.(2.37) into:

$$Z_{ij} = X_{ij}W^V \odot a_{ij} \tag{2.39}$$

Where W^V is a weight matrix that is trained along side the model and allows to control the world representation size (Z_{ij}). Both variations do not contribute significantly to the

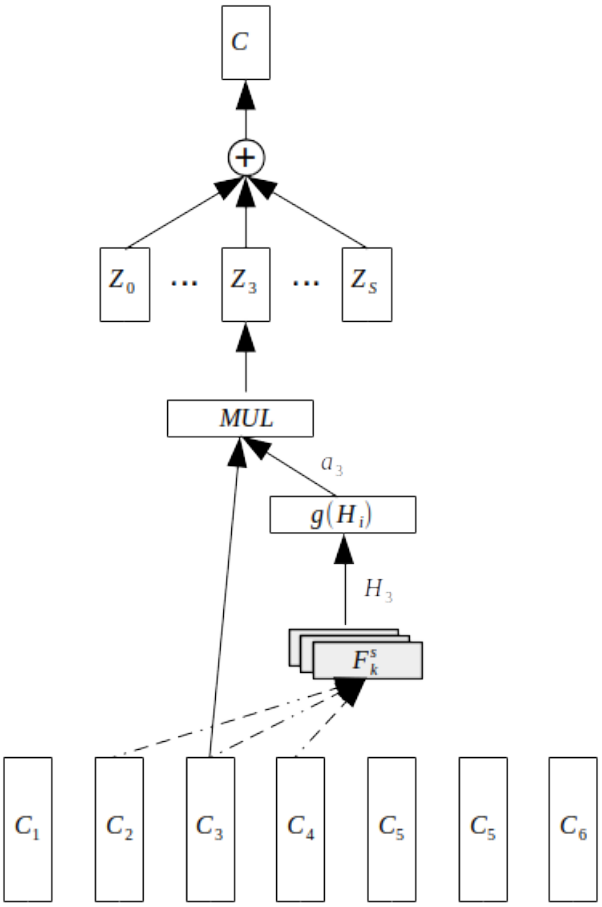

Figure 2.26 – Illustration of the convolution-based attention mechanism developed in our model at document level.

improvement of the performance of our main model as the model needs further tuning and experimentation. We do not report any further the experimentation we have carried out using these variants of our model.

2.3.3 Experimentation

In order to benchmark our two contributions against the state of the art methods, we used five data sets to cover a relatively wide range of document topics, various scales (sizes of the datasets) and tasks. These data sets are listed below with a description of their characteristics:

1. 20NewsGroup [55], that we will be denote as 20NG, is an English-language textual corpus of 20K documents containing messages, comments and news collected from a discussion forum. This dataset is categorized into 20 categories. The documents contain noise in the form of meta-data, that is particularly present in the headers and footers of the messages. This noise contains information about the authors' name and affiliations. These meta-data can sometimes be beneficial to a classification task in some cases by biasing the learning.
2. RCV1 [56] is a corpus of 800K documents containing English-language news bulletins collected by the Reuters news agency, and categorized into 103 categories. The task is *multi-label*, in other word a document can be labeled with several categories.
3. TED-FR is a sub-corpus from the fd-TED corpus [7] [6]. This corpus contains tenders from the European public procurement. The tender documents contain a lot of noise that comes in the form of legal-commercial-administrative information. The documents are categorized into 45 categories and are also *multi-labeled*. More precisely, the fd-TED dataset is organized according to a hierarchical category architecture: we use for our experimentation the level 2 hierarchy which leads to the 45 categories mentioned above. For this experiment, we only consider the calls for tender in French language that are fully translated (from the 24 languages of the European union), which constitutes 800K documents.
4. TED-FILTER, is a corpus comprising 2000K documents extracted from the complete fd-TED (the corpus is multilingual but we only consider the document in

6. `https://github.com/oussamaahmia/TED-dataset`

french) corpus by filtering out the legal-commercial-administrative paragraphs. This filtering step considerably reduces the size of the texts (around 80% of the document). The documents are still categorized into 45 categories in a *multi-label* setting.

5. OHSUMED [57] is a subset of the MEDLINE database, which is a bibliographic database of peer-reviewed medical literature (in English) maintained by the National Library of Medicine. It contains 22K documents divided into 23 categories presented in a *multi-label* setting.

Experimental protocol

We describe in this section the protocol we used to evaluate our proposed document embedding methods (combination W2V+LSA and CnHAtt) comparatively to the state of the art approaches reported in the literature. All the five datasets briefly described above are used for the evaluation.

The following pre-processing is applied to all tested methods :

We first remove the stop-words (we use the stop word lists provided by the NLTK toolkit [58]) corresponding to the languages that are used in the datasets. Then, for the TED-FR corpora, the CPV codes (classification system for public contracts) contained in the text are replaced by a neutral keyword (%digit%), as these codes define the classes of the documents. In addition, we also ignore terms whose number of occurrences is less than 5 in a dataset.

Once the previous preprocessing is done, we consider the following vectorizations (embeddings) methods:

1. *tf*: bag of words with a *tf* weighting
2. *LSA tf*: *tf* weightings + LSA
3. *tf-idf*: bag of words with a *tf* weighting
4. *LSA tf-idf*: *tf-idf* weightings + LSA
5. *W2V* the average word2vec vectors
6. *LSA+W2V* the average word2vec vectors concatenated with the LSA vector using a *tf-idf* weighting

In our experiment, we have choosed 100 dimensions for LSA and W2V which leads to a combined vector of 200 dimensions for *LSA+W2V* vectorization.

For our hierarchical model (CnHAtt) and HAT [11] model the embedding layers are initialized using a 100 dimension word2vec embedding.

Beside CnHAtt and HAT we have evaluated each vectorization method described above using the following classifiers:

— MLP: A multilayer perceptron [59] with two hidden layers of 200 nodes each, the activation function used is the sigmoid.
— SGD: A linear support vector machine optimized by stochastic gradient descent [60].
— NB: A multinomial naive Bayes classifier [61] with $\alpha = 1$ for the *tf* and *tf-idf* weightings. A Gaussian naive Bayes classifier [62] is used for the other vectorizations, as they can contain negative values, with a default variance of $\sigma = 10^{-9}$.
— for CnHAtt and HAT models the outputs are connected to MLP layers that will be trained along side each model (the other classification models cited above are not used in this case). Both CnHAtt and HAT models use the same fixed number of words and sentences, namely 50 and 20 respectively. For CnHAtt we have used 128 filters on both CNNs with a window size of 5 and the GRU layer in the HAT model includes a hidden layer whose size is 100 neurons.

However, not all combinations of vectorization/classification model can be evaluated for algorithmic complexity reasons. It is indeed difficult to combine, for example, a very large (high dimensional) vectorization *tf-idf* with an MLP classifier for instance.

The '*multi-label*' classification was done using the '*One v.s. Rest*' classification framework (in the case of MLP this step is not required), which involves training a binary model for each class. Observations (documents) belonging to one class are considered positive class of an observation where all others as negative class (we end up with binary classifiers for each class).

A decision threshold varying between [0.1,..,0.9] [7] is then used: if the log-probability of a tested document in relation to a class is greater than the threshold, the tested document will be classified as member of this class.

7. we keep the threshold with the best accuracy score for each classification model

2.3.4 Experimentation Result

Using word2Vec averaging as document embedding for the different NLP tasks (classification, clustering, recommendation, etc.) that we face at OctopusMind, was indeed a good start: it was fast and easy to implement in our production environment. However, this approache rapidly shows some limitation and disadvantages.

We first try to verify the general assumption that word2vec averaging tends to focus on the main topic of the document and to filter out some important semantic information while storing only the general idea of a document. In addition, we want to evaluate the 'noise' produced by the way contextual information is used to construct word embedding: the word that are frequently in a similar context will inevitably have similar embedding vectors even if these words are critical to dissociate two different classes (antonyms, such as *to love* or *to hate* are generally associated to similar embedding vectors since they occur frequently in similar contexts).

After building a skipgram word2vec model on the 20newsgroup dataset, we took, as an example, the words "Jesus-Christ" and "Moses" (two Prophets). These nouns tend to occur in a similar context and after calculating the cosine similarity between the word2vec vectors of this two words we obtained the value of (0.80) which is fairly large considering that this two words are really important to separate classes like: Judaism and christianism or ("soc.religion.christian" and "talk.religion.misc") in the case of 20newsgroup. To get a better view on this problem, we can analyse the confusion matrix produced by a SGD classifier and the three different embeddings (W2V, LSA, LSA+W2V). The results are shown in the table below:

embedding	alt.atheism	soc.religion. christian	talk.politics. mideast	talk.religion. misc
W2V	23	16,25	3,4	59,45
LSA	21,33	11,15	**1,11**	76,08
W2v+LSA	**16,43**	**8,58**	1,69	**84,5**

Table 2.1 – Excerpt of the confusion matrix for talk.religion.misc category obtained using a SGB classifier on the 20NG dataset, using the various embeddings.

From table 2.1 when we use a word2Vec averaging embedding, the classifier tends to misclassify the documents from *alt.atheism* and *soc.religion* into *talk.religion.misc*. Using a LSA embedding reduces the confusion between these classes that are close semantically, and the combination of LSA and word2vec embeddings improves the results even more.

The following tables present the experimental results obtained for the different combinations of vectorization and classification algorithms.

The classification algorithms are evaluated using four metrics:

— The empirical accuracy (*accuracy*) which represents the percentage of the observations labeled identically to the true annotation:$A = \frac{vpvn}{vp+fp+vn+fn}$,

— The precision measure $P = \frac{vp}{vp+fp}$,

— The recall measure $R = \frac{vp}{vp+fn}$ where vp is the number of true positives, fp is the number of false positives and fn is the number of false negatives,

— The F1-measure $F = \frac{(1+\beta^2).P.R}{\beta^2.P+R}$ (with β = 1),

The precision, recall and F1-score measures are presented in the form of a weighted average relative to the support of each class. It is therefore a macro-average, that allows to take into account the imbalanced classes.

In addition, 80% of each dataset is used as training data and 20% as test data, enabling to set up a k-fold cross-validation with $k = 5$.
As stated above, we note that for neural network methods we apply different threshold on the outputs and we select the best results on a validation set based on the accuracy (thus, it is possible to have different threshold for each step of the cross validation), the different metrics for each step of the cross validation are then averaged.

The tables (2.2, 2.3, 2.5 and 2.6) present the classification results obtained on all five corpora.

For the 20NG corpus, the best accuracy is obtained by the SGD classifier using the *tf-idf* bag of word embedding. This can be explained by the fact that *tf-idf* weighting is able to represent documents by keeping all the occurrences of the words contained in a document, which allows to separate more efficiently the most similar classes of this very peculiar dataset: 20NG is a dataset that contains very similar classes (semantically speaking), i.e: religion.misc and alt.atheism.

	Accuracy	Precision	Recall	F1-score
SGD (tf-idf)	92,92	92,8	92,8	92,8
CnHAtt	90,29	96,38	94,19	90,63
NB (tf-idf)	90,07	90,6	90	89,8
HAT	88,24	94,37	92,30	88,72
MLP (LSA+W2V)	84,46	84,6	84,6	84,6
NB (tf)	83,96	86,2	83,8	83,4
MLP (LSA tf-idf)	81,86	82	81,8	81,6
SGD (LSA tf-idf)	78,99	81,2	79	79
NB (LSA tf-idf)	74,1	75,4	74,4	74,4
MLP (W2V)	73,43	73,6	73,6	73,6
SGD (LSA+W2V)	71,49	80,8	71,4	73
SGD (tf)	71,12	83,2	71	74,6
NB (LSA+W2V)	68,46	71,6	68,4	69
MLP (LSA tf)	59,37	59,2	59,4	58,4
SGD (LSA tf)	54,16	68,4	54,2	54,4
NB (W2V)	51,95	55	51,8	52
SGD (W2V)	51,16	65	51,2	52,6
NB (LSA tf)	39,23	51,2	39,2	41,6

Table 2.2 – Results obtained for the 20NG dataset.

Concerning all four other datasets, our LSA+W2V model gets relatively good results for a cheap computational cost (using MLP classifier).

The relatively low results obtained on the TED-FR dataset by the LSA+W2V embedding method, even though the recall is significantly higher than the one obtained with the traditional tf and tf-idf BOW models, are largely due to the noisy nature of the database, in particular because it contains a fairly large amount of legal-administrative information (which represents the majority of the texts). Therefore the discriminating words (relevant) in relation to some classes represent only a small portion of the text.
In another hand, this is precisely the reason why attention based vectorization models perform much better, as compared to LSA+W2V vectorizations (or any other methods that do not include any attention mechanism) that take into account all the words of a document without any discrimination regarding their importance to the classification. Conversely, the attention based methods (CnHAtt and HAT) only stores the most important words contextually to the task at hand. The hierarchical architectures also helps when working with relatively long documents (RCV1 and TED-FR)

Furthermore, from tables 2.4 and 2.2, we observe that the performance of the LSA+W2V vectorization is relatively close to CnHAtt and HAT vectorizations when working with relatively clean and small datasets (example in 20NG we obtain f1-scores of: 90.63, 88.72, 84.6 for CnHAtt, HAT and LSA+W2V respectively).

Concerning TED-FILTER, hierarchical method was not evaluated, the reason is that this dataset contain very small texts (one sentence in most cases). Therefore the hierarchical aspect of HAT and CnHAtt cannot be highlighted on such dataset.

From the tables (2.6, 2.4, 2.5), we can infer that by combining LSA with word2vec, our model, thanks to LSA, takes into account the frequencies of the important words that compose the document and, thanks to the averaged word2vec embedding, this kind of vectorization takes into account the general semantics of the documents, while preserving a reduced dimension for the resulting vector. By taking advantage of the complementary of the two approaches, LSA + W2V achieves a good compromise between performance and algorithmic complexity as shown for the RCV1 and TED datasets.
For 20NG, which has very similar and ambiguous categories, LSA vectorization is probably carried out with an insufficient dimensionality, which prohibits a good separation of the words that characterize most these categories.

	Accuracy	Precision	Recall	F1-score
CnHAtt	84,89	99,49	88,95	89,74
HAT	84,40	99,13	82,14	89,25
MLP (LSA+W2V)	55,27	78,2	57	64
MLP (LSA tf-idf)	49,65	76,8	49,4	55,4
SGD (tf)	48,32	78,2	59,6	62,8
MLP (W2V)	45,23	76,6	45	53
MLP (LSA tf)	33,13	64,4	31,8	37
SGD (LSA+W2V)	32,8	66,6	34,4	41
SGD (W2V)	29,9	64,4	31,8	38,4
SGD (LSA tf-idf)	17,63	53	16,6	22
NB (tf)	16,65	34,6	81,2	45,6
SGD (tf-idf)	16,35	73,8	15	20,4
NB (LSA tf-idf)	14,6	32	45,6	35,2
NB (tf-idf)	13,99	68,8	13,4	19,8
NB (LSA tf)	11,41	28,2	41,8	30,4
NB (LSA+W2V)	6,08	26,4	60,8	33,2
SGD (LSA tf)	6,06	30,2	5,8	9
NB (W2V)	5,33	19,8	48,2	25,6

Table 2.3 – Results obtained for the TED-FR dataset.

	Accuracy	Precision	Recall	F1-score
CnHAtt	78,12	92,99	92,25	92,57
HAT	77,08	92,16	91,52	91,81
MLP (LSA+W2V)	72,58	89,36	90,05	89,66
SGD (tf)	71,45	87,98	90,25	89,06
MLP (LSA tf-idf)	70,88	88,47	89,50	88,94
MLP (W2V)	69,25	87,48	87,52	87,44
SGD (tf-idf)	64,51	93,28	84,40	88,51
MLP (LSA tf)	50,95	84,01	77,13	80,23
NB (tf)	25,80	62,87	90,09	73,31
SGD (LSA+W2V)	19,64	71,96	49,67	56,84
SGD (LSA tf-idf)	19,47	74,52	43,34	53,33
NB (tf-idf)	15,25	88,64	34,20	44,55
NB (LSA tf-idf)	12,94	54,46	53,57	53,33
SGD (W2V)	11,82	63,40	34,42	41,61
SGD (LSA tf)	10,25	69,34	25,13	34,68
NB (LSA tf)	3,29	33,28	47,72	37,13
NB (LSA+W2V)	3,15	38,56	66,54	47,25
NB (W2V)	1,31	32,03	61,63	40,39

Table 2.4 – Results obtained for the ohsumed dataset.

	Accuracy	Precision	Recall	F1-score
CnHAtt	64,60	89,62	85,30	86,93
HAT	63,15	89,96	83,45	85,37
MLP (LSA+W2V)	58,51	87	81	83,6
MLP (W2V)	57,86	87,4	80	83
MLP (LSA tf-idf)	54,64	86	77	80,4
SGD (tf-idf)	49,67	91,4	69,2	76,2
MLP (tf)	49,41	84,2	72,2	76,6
SGD (tf)	44,84	81,6	75,6	77,8
SGD (LSA+W2V)	38,32	79	71	73,4
SGD (LSA tf-idf)	36,42	82,8	60,6	67,2
SGD (W2V)	32,99	76,6	67,4	70
SGD (tf)	18,81	78,4	44	53
NB (LSA tf-idf)	4,85	40,8	77,2	50,2
NB (LSA+W2V)	2,58	40,6	85,2	50,4
NB (W2V)	1,54	38,6	85,2	48,4
NB (tf)	0,55	27,4	72,8	36,2
NB (tf)	0,29	4,4	0	0
NB (tf-idf)	0,29	0	0	0

Table 2.5 – Results obtained for the RCV1 dataset

	Accuracy	Precision	Recall	F1-score
MLP (LSA+W2V)	77,47	95,2	75	83,2
MLP (W2V)	75,6	95,2	72,6	81,4
MLP (LSA tf-idf)	69,38	94,4	66,4	76,6
NB (tf-idf)	66,07	88	66,2	74,6
SGD (LSA+W2V)	61,22	91	59,2	69,8
SGD (W2V)	56,16	88	55,2	65,6
SGD (tf)	55,74	89,4	53,8	65
NB (tf)	51,9	62,8	78,2	68,8
SGD (tf-idf)	49,39	93,4	46	59
SGD (LSA tf-idf)	44,73	83,2	42,4	53,4
SGD (tf)	31,82	74,4	30,2	39,2
NB (W2V)	5,72	23,8	77,6	33,8
NB (tf)	2,38	19,8	73	27,8
NB (LSA+W2V)	1,85	22,8	81,2	33,8
NB (LSA tf-idf)	1,63	20	76,8	30,2

Table 2.6 – Results obtained for the TED-FILTER dataset.

We also observe from the previous tables that CnHAtt and HAT seems to have very close results for all datasets with CnHAtt slightly ahead. The main difference between the two approaches lies in their respective algorithmic complexity. In addition, CnHAtt is fully parallelizable which is not the the case for HAT since it is based on RNNs (GRUs in the case of this experiment) that is intrinsically a sequential model. The following table 2.7 presents the execution time for both models on a batch of 512 items. The hardware used for this experiment has the following characteristics: (CPU: 2 x Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz, GPU: GeForce RTX 2080 Ti).

	CnHAtt	HAT
Training	51 ms	498 ms
Prediction	17 ms	118 ms

Table 2.7 – Average time needed by CnHAtt and HAT to train and predict a batch of 512 observation.

Clearly CnHAtt is more efficient computationally speaking comparatively to HAT: CnHAtt is nearly 10 times faster than HAT, which represents an order of magnitude.

In order to evaluate how our model CnHAtt performs comparatively to the HAT model in clustering tasks, we carried out the following experiment. For each one of the datasets listed in (2.3.3), we only consider the documents that belong to only one class (while filtering out documents belonging to several classes). The HAT and CnHAtt models are then trained on each dataset (the training is stopped when the loss on the validation dataset does not change for more than 3 epoch)

The vectorization models are evaluated using Kmeans [63] and Hierarchical clustering [64] with cosine similarity, the number of expected clusters for each dataset is the same as the number of classes. The performance is then measured using the Rand Index[65] measure. The results are presented in the following tables.

	CnHAtt	HAT
20NewsGroup	0.50	0.48
RCV1	0.45	0.43
TED-FR	0.46	0.61
ohsumed	0.25	0.16

Table 2.8 – Results obtained for clustering using Kmeans algorithm.

	CnHAtt	HAT
20NewsGroup	0.38	0.37
RCV1	0.33	0.30
TED-FR	/	/
ohsumed	0.29	0.21

Table 2.9 – Results obtained for clustering using Hierarchical clustering algorithm. The evaluation measure is the Rand index (No evaluation has been obtained for TED-FR due to the size of the data set).

From table (2.8), we observe that in some cases the score is surprisingly high (HAT on TED-FR). This is probably caused by the fact that document vector becomes too much related to the output categories. To have a better look at the results and to validate this assumption, we have used the document vectors trained on TED-FR (45 classes) and tried a clustering on the TED-FR with level 3 hierarchy classes (317 classes). Using the Rand index we get 0.23 and 0.17 for CnHAtt and HAT respectively using the Kmeans algorithm.

That confirms that the document generated by HAT model tends to be too much related to the classes. This may be caused by the way HAT generates the document vectors, using a GRU which is a non linear projection of the word embeddings, causing the document vector to live in a different space. Conversely, CnHAtt involves a weighted sum of the word embedding exploiting the attention weight, which means that the document vector lives in the same space as the word vectors. Also we observe heuristically that only a small modification of the embedding is produced during the training phase at the embedding layer.

This mean that the document vectors generated by CnHAtt live in the same space as the word vectors which is not the case for the HAT model.

The following figures illustrate the attention generated by the CnHAtt model, using green shades at word level and blue shades at sentence level (the darker the color, the bigger the attention). We notice from figures (2.30, 2.29) that the model gives less attention to sentences that contains legal-commercial-administrative information, which shows that the CnHAtt model is able to work with noisy documents.

On the figures (2.28, 2.27), we show that the same word can have different attention levels according to its context.

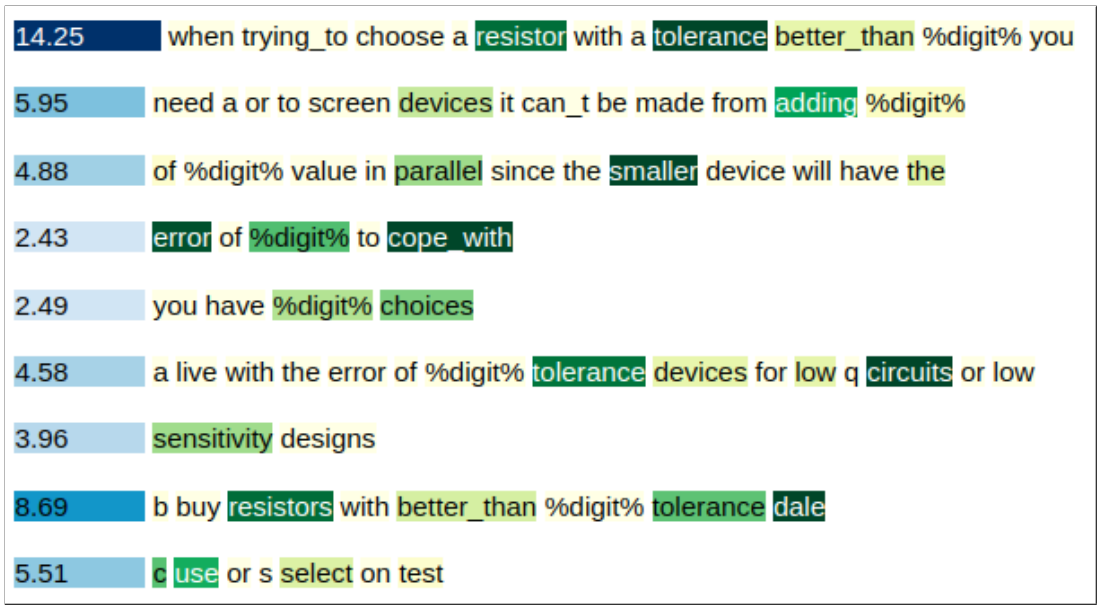

Figure 2.27 – Figure illustrating attention weights in a sentence and a word level on a 20NewsGroup document in the class "sci.electronics"

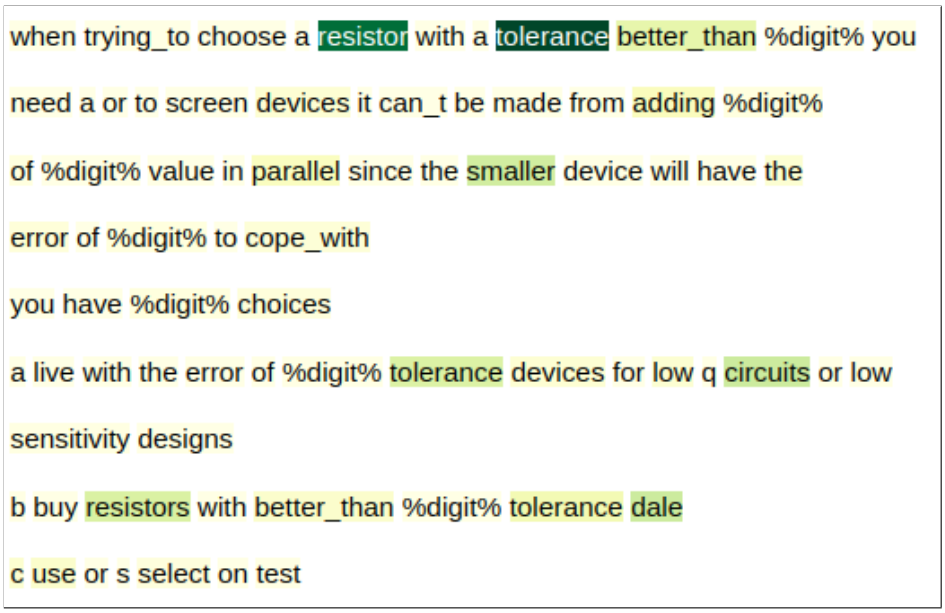

Figure 2.28 – Figure illustrating attention weight for a word by multiplying the word attention weight by sentence attention weight on a 20NewsGroup document in the class "sci.electronics"

Figure 2.29 – Figure illustrating attention weights in a sentence and a word level on a FULL-TED document in the class "Installation services of communications equipment"

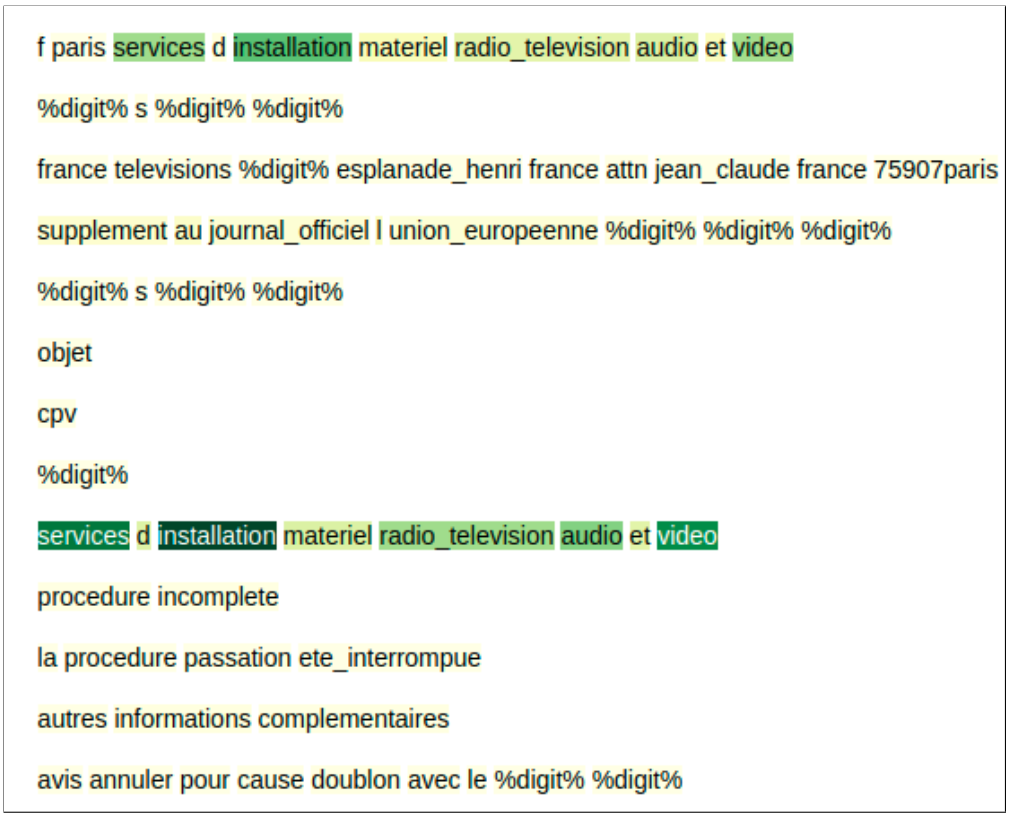

Figure 2.30 – Figure illustrating attention weights a word by multiplying the word attention weight by sentence attention weight on a FULL-TED document in the class "Installation services of communications equipment"

2.3.5 Conclusion and perspectives

In this chapter we described the two document embedding methods we developed, namely LSA+Word2vec and CnHAtt. both models gave promising results for classification tasks.

As for LSA+Word2vec by using the complementarity of Word2vec and LSA we were able to create dense document vectors that stores the context of word occurrences (local context for Word2vec and global context for LSA), the document vectors obtained are of a low-dimension, therefore respecting the criteria set by OctopusMind.

The CnHAtt model, is a CNN based attention DNN. The model brings a novel way to produce local attention weights using CNNs, it also applies a hierarchical architecture, allowing to produce attention at a word level giving each word a weight according to

its local context and on a sentence level by giving each sentence a weight according to their relevance to classification task, which allows to remove irrelevant sentences (legal and administrative sentences in the case of public procurement documents). The CnHAtt model do not use RNNs, allowing it to be 10 times faster to train compared to other state of the art RNN based attention models. Also the CnHAtt model beats the state of the art similar model in the presented classification tasks.

As a perspective, we will be working toward updating the CnHAtt model to work with multiple languages. For the time being in order to achieve document classification on multilingual corpus, we train a different model for each language. Also, we need to further study the different variation of CnHAtt on different NLP problems.

In addition to that we will be exploring more clustering techniques, with the goal of being able to detect new trends in public procurement (new activity fields for example), keeping in mind that due to the nature of the corpus we are working with, the clustering methods we will be using will have to take into consideration the nature of the data we are working with. Especially the fact that the documents we are willing to classify may belong to several clusters. Fuzzy clustering methods[66], [67], seems to to be a good answer to this problematic, we are aiming to explore these techniques in the foreseeable future.

CHAPTER 3

APPLICATIVE TASKS

3.1 Introduction

The main objective and purpose of this thesis project is to provide a toolbox that performs automatic strategic monitoring on call for tender databases using natural language processing.

In order to achieve such a goal we have to create software modules that perform a variety of tasks ranging from specific information extraction from noisy document to modeling complex semantics of raw text, as well as tunneling the right information to the right actor.

Those different solutions must fit in an already existing software environment in use at OctopusMind and has to meet the constraints imposed by the company (time complexity, cost...).

We decided to take a modular approach, in a sense that we created a dedicated solution for each task: the developed modules have been created in a way that there are easily reusable for similar problem. The combination of such relatively simple modules should be able to solve more complex problems, with the hypothesis that each complex problem can be split into simpler problems.

This chapter focuses on describing the different solutions that we have developed and deployed to solve practical company problems, by either applying state of the art methods or developing our own solution to fit the specific needs of the company.

This chapter is decomposed into three sections. The first section describes the solution related to the information and relationships extraction problems (covering financial data extraction, surfaces detection, etc.). The second section is about the creation of a recommender system using different document embedding methods described in the chapter 2. Finally we list the future works and perspectives that are planned in order to tackle more complex problems and overcome some of our shortcomings.

We also note that the different problems tackled in this chapter were dealt with

all along the thesis period (4 years period) and in parallel to the works on document embedding.

3.2 Information extraction

The data found on the web, can be categorised in two main classes: structured data and unstructured data.

Structured data is characterized by a "standardized" format that can be easily exploited allowing a direct access to the relevant information (SGBD database, xml data, rdf data). On the contrary, unstructured data does not have a pre-definded formatting. hence, it contains information that cannot be immediately accessible, and generally requires human expertise to be properly processed.

The unstructured information is estimated to represent about three quarters of the data accessible on the internet [68]. The data used by OctopusMind, and the public procurement documentation in general, falls into this category: the documents contain large amount of useful information that are embedded into the full-text. Examples of unstructured information include:

— "Financial data", which requires detecting and extracting information about the cost of a project, the budget allocated or the minimum turnover needed to apply for a call for tenders.

— "Renewal information", which requires extracting from a call for tender information that allows to know if a given project will be renewed and if so, when this will occur.

— "Surface characterization", which requires in analysing documentation related to construction projects and extracting different information about the building that are described (surface, type,...)

Creating automated tools to automatically extract such information became quickly a necessity, since, as the number of documents grows, manually processing them became impossible.

A lot of work in the literature has provided numerous models for information extraction (e.g. named entity detection) and relationship extraction. These methods are generally grouped into two categories: supervised approaches and unsupervised approaches [69].

Unsupervised approaches are usually based on contextual features. Distributional

semantics introduced by Z.S Harris [70], assumes that words (entities) that frequently occur in same contexts, tends to have similar meanings. D. Ravichandran uses a bootstrap approach [71] to learn surface text patterns in order to extract binary relationships from the Web. More recently [72] developed a large scale method for relationships extraction that allows handling polysemy and synonymy problems, usually source of semantic ambiguity and is in general difficult to overcome for the extraction of certain classes of relationships.

This approach is divided into two steps. The first step consists in discovering a set of semantic classes that will be used as arguments for each binary relation.

The result of this first step is a large collection of relations whose arguments are pairs of semantic classes associated to a single semantic relation. These type of relationships are called "Type-A" relations. Example:

<{New York, London...}, be locate in, {USA,England,...}.>

During this phase, the polysemous phrases are disambiguated and placed in separate relations (Example: <Euro, be the currency of, Germany> and <authorship, be the currency of, science> are polysemous relations).

The second phase groups the "Type A" relationships according to their similarities [73], this is done by Hierarchical Agglomerative Clustering [74] resulting in "Type B" relationships. It results in relationships that can have multiple expressions (semantically similar sentences written in different ways), Example:

$r1 =$ <Cities, be locate in, Countries> and
$r2 =$ <Cities, be city of, Countries>.

Therefore being able to resolve hyperonyms [1] problem (Example: the relations $r1, r2$ share a pair of arguments $< Tokyo, Japan >$ and a pair of hyperonyms $< City, Country >$).

The supervised methods address relationships extraction as a classification problem. Usually supervised approaches are divided into 2 sub-groups: kernel based methods and features based methods.

The kernel-based method tackles the problem of relationships extraction as a kernel evaluated on a pair of objects (binary matching) [76], which allows the use of large feature sets without needing to extract them explicitly. This model exploits a kernel based on the syntax tree of a sentence. Qian et al [77] developed a dynamic approach that

1. hyperonym is a word with a general meaning that has basically the same meaning of a more specific word. [75]

determines sub-trees that could potentially encode a relationship, by using a convolution tree kernel. More recently Zhou et al [78] proposed an approach that uses enriched syntactic and semantic information to develop a context-aware convolutional kernel (-sensitive convolution tree kernel). This type of kernel is used to extract sub-trees that encodes relationships that take into account the context.

Features based methods seek at extracting a set of syntactic and semantic variables that are then used in a statistical learning framework [79]. Several techniques have been proposed in this line of research, in particular structured classification algorithms such as CRF [80] (Conditional Random Fields). More recently a semi-supervised approach has been proposed in [81] for extracting relationships from large corpus.

We also note, that with the emerging of deep learning techniques, many attempts have been done to use deep learning architectures in named entity detection and relationships extraction. Either by using CNNs (Convolutional Neural Networks) like in [82] where multi-level attention CNNs are configured to classify relationships by relying on the two levels of attention: the first one is used to to capture entity-specific attention (at input level in respect to the target entities) and the second one comes in the form of an attention based pooling layer that calculate relation-specific attention, enabling it to automatically learn which parts are relevant for a given classification. Other approaches take advantage of RNNs (Recurrent Neural Networks) like in [83] where the LSTM (Long-Short Term Memory) network is trained to classify relationships between named entities. The main feature of this model is the fact that it chooses the shortest dependency path, thus only retaining the most relevant information (to relation classification task).

For our needs we will focus on statistical feature based methods. The main motivation behind this choice is that most of the problems we tackle are centred on detecting specific relationships/entities that we know beforehand (we mostly deal with supervised entity and relationships detection) so we have no need to explore the corpus in search of unknown relationships (unsupervised entity and relationships detection). The use of kernel based methods has not been considered as they are essentially used for unsupervised entity-relation detection. Furthermore, we have not considered either deep neural methods, since such models need relatively large dataset to be trained, which are costly to create. Also this type of models cannot be easily interpreted and the model behavior in most cases are impossible to explain.

The models we have used for the different information tasks are: structured perceptron, HMMs (Hidden Markov Model), CRFs (Conditional random fields), HCRFs (Higher order CRFs) and Semi-Markovian CRFs. These models are the most used models for feature based information and relationships extraction. We evaluated them on the surfaces extraction task, to select the best model as a starting baseline for other similar tasks, namely: financial data and renewal information extraction.

We also developed a context aware version of bag of word, for financial information extraction. Which is faster than the previously quoted models.

3.2.1 Models used

Conditional random fields

Conditional random fields (CRFs) [84], are a class of probabilistic modeling method, used for labeling and segmenting structured data (such as sequence or tree). Its main objective is that of defining a conditional probability distribution over a label sequences y, given a sequence observation x.

The Conditional Random Fields can be defined as follows: Let $G = (V, E)$ be an undirected graphical model, where V is the nodes set (Vertices), E is the edges, X and Y are two random fields describing respectively the observations and the labels, in such a way that for each node $v \in V$, $\exists Y_v \in Y$. We can assume that (X, Y) is a Conditional random Field if each random variable Y_v satisfies the following Markovian property :

$$\forall v, p(Y_v|X, \{Y_w, w \neq v\}) = p(Y_v|X, \{Y_w, v \sim w\})$$

, where $v \sim w$ means that w and v are neighbors in G (i.e. there is an edge linking w and v).

In other words, each random variable Y_v only depends on X and on its neighbors in G (a more intuitive way to say it is that the prediction of the label depends on the previously predicted label).

The potential functions associated to the CRF are defined as the exponential of a weighted sum of functions f_k (referred to as "feature functions"), the weights being the λ_k. The conditional probability of a label given the knowledge of an observation x, can be formulated according to the following equation:

$$p(y|x_{1:N}) = \frac{1}{Z(X=x)} \exp(\sum_{n=1}^{N} \sum_{i=1}^{F} \lambda_i f_i(z_{n-1}, x_{1:N}, n)).$$

Where: $Z(x)$ is a normalisation function specific to the observation x:

$$Z(x) = \sum_{y} \exp(\sum_{n=1}^{N} \sum_{i=1}^{F} \lambda_i f_i(z_{n-1}, x_{1:N}, n)).$$

Higher order CRFs and semi-markovian CRFs

A CRF is considered as being a high order model if each Y_i depends on a fixed number o of previous labels Y_{i-o},...,Y_{i-1}. The training and inference (prediction) for this type of models is only practically possible for small values of o [85].

Unlike the CRF that allows for a given sequence of observation $X = x_1, x_2, ..., x_T$ to return a sequence of labels $Y = y_1, y_2, ..., y_T$ that maximise the *posterior probability*.

The semi-Markovian CRFs allows to label sub-sequences (segments) of the observation $S = \langle s_1, ..., s_p \rangle$, where the segment $s_j = \langle t_j, u_j, y_j \rangle$ has a start position t_j, an end position u_j and a label $y_j \in Y$.

For example, for $Y = \langle 1, 1, 2, 2, 2, 0, 0, 0 \rangle$ we have $S = \langle \langle 1, 2, 1 \rangle, \langle 3, 5, 2 \rangle, \langle 6, 8, 0 \rangle \rangle$.

In the case of semi-Markovian CRFs the features are calculated at segment level rather than at an element level. Furthermore, for the "linear chain" CRF, a feature of a segment j, lets say for example the n-th feature of the segment j, $f_n(j, X, S)$, is calculated from the observations $x_{t_j}, ..., x_{u_j}$.

Hidden Markov Model

Hidden Markov Model (HMM)[86] is composed with a set of finite states connected through transitions. Each state is characterized by two types of probabilities: the probability of transitioning between states and the probability of emitting a label, which can be a discrete probability function or a probability density.

A HMM can be defined as follow: Lets S be a set of states including an initial state S_1 and a final state S_f, A a transition probability matrix (such as $A = a_{i,j}$ where a_{ij} is the transition probability from the i-th state to the j-th state) and B is the output probability matrix $B = \{b_j(O_k)\}$ (in the case of a discrete form of HMMs) where O_k in this case is a discrete symbol.

In the discrete HMM form, a_{ij} and $b_j(O_k)$ have the following properties:

$$a_{ij} \geq 0, b_j(O_k) \geq 0, \ \forall i, j, k,$$

$$\sum_j a_{ij} = 1 \ \forall i,$$

$$\sum_k b_j(O_k) = 1 \ \forall j.$$

Thus, A gives the probability of transition between states and B associate each state Sj, said hidden, with a probability for each output (the observable state).

The figure 3.1 illustrate diagram of an HMM along side a CRF.

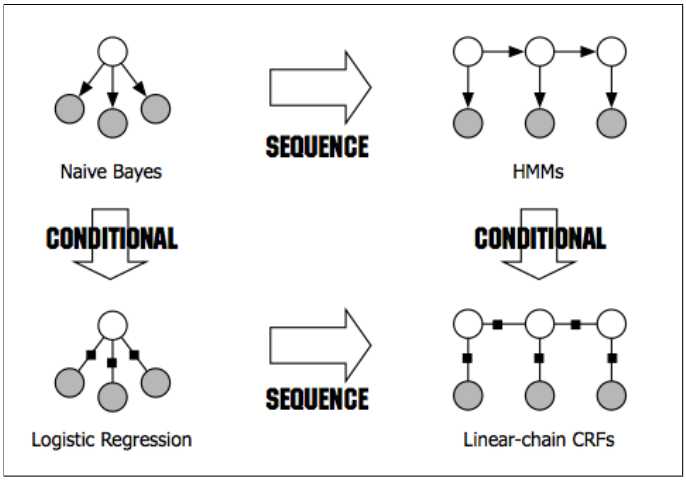

Figure 3.1 – Diagram of the relationships between naive Bayes, logistic regression, HMMs and linear-chain CRFs taken from[87]

Structured perceptron

Structured perceptron [88] is the result of combining the percepton algorithm[89] with an inference algorithm.

The perceptron enables a local linear classification, while the inference algorithm (Viterbi algorithm[90]) allows to align sequential data. The model can be summarized as follow: $\phi(X, y)$ links an observation X to a label y, θ is the weights vector and GEN is the function that performs the prediction.

For each iteration:
  for each (X, y):
    $y* = argmax_{y \in GEN(x)} \theta^T \phi(x, y)$
    if $(y* \neq y)$ then: $\theta = \theta + \phi(x, y) - \phi(x, y*)$

3.2.2 Surface extraction

Urban planning is a complex area where public decision-makers intervene. Indeed, each development project affects and interacts with the other projects in a given area. Which makes anticipating changes and analyzing their long-term impacts a very complex task. These tasks require a comprehensive knowledge of the urban environment and future projects. It is in this context that the identification of future territorial development projects (Urban planning) and all information related to them becomes a necessity. As is, it allows to represent the future evolution of a city, and for this purpose, it is necessary to get information on the nature of the development project at the right time. This information are characterized in particular by its surface area and its main type, characterized by a building typology: hospital, park, etc.

The main purpose of this work is, from the content of call for tenders documents, to list different development projects (public buildings construction) in France, and extract their surfaces, location (Geo-position), in order to plot them on a 3D map.

Methodology

Our problematic is to extract the surface of a territorial development project (a building or a group of buildings) based on unstructured textual data. More concretely we need to extract two types of named entities (surface and building) from unstructured text, to determine if there exists a relationship between them.

Our problem is therefore divided into two steps: Firstly, the named entity detection (surface and building). And secondly the relationship extraction of the type **"is a surface of"** between the previously detected entities. These relationships link a building to it's surface, like for example "a **1000**m^2 **municipal swimming pool"**. The relationships can also link several buildings to one surface, for example: Approximately **2500 m2** of floor space, including **offices**, **reading rooms** ...".

The proposed approach is to perform the named entities extraction (surface and building) simultaneously with the relationships detection between these entities. This

approach can be seen as a problem of **sequence labeling** or "**sequence tagging**", with the following possible labels:

- **O**: Items that do not fall into neither of the following categories. ;
- **LINK_SURFACE/bâtiment**: the buildings that are part of a relationship **" is a surface of"**;
- **O/bâtiment**: Buildings that are not part of a relationship **"is a surface of"**;
- **O/surface**: surfaces that are not part of **"is a surface of"**;
- **LINK_SURFACE/surface**: the surfaces that are part of a relationship **" is a surface of"**;
- **LINK_SURFACE**: Items that are part of the **"relationship are a surface of"**, but are neither surface nor building (this label makes it easier to link the named entities together in the case that there are several surfaces and buildings in the same sentence).

The figure 3.2 shows the expected result of our labeling model.

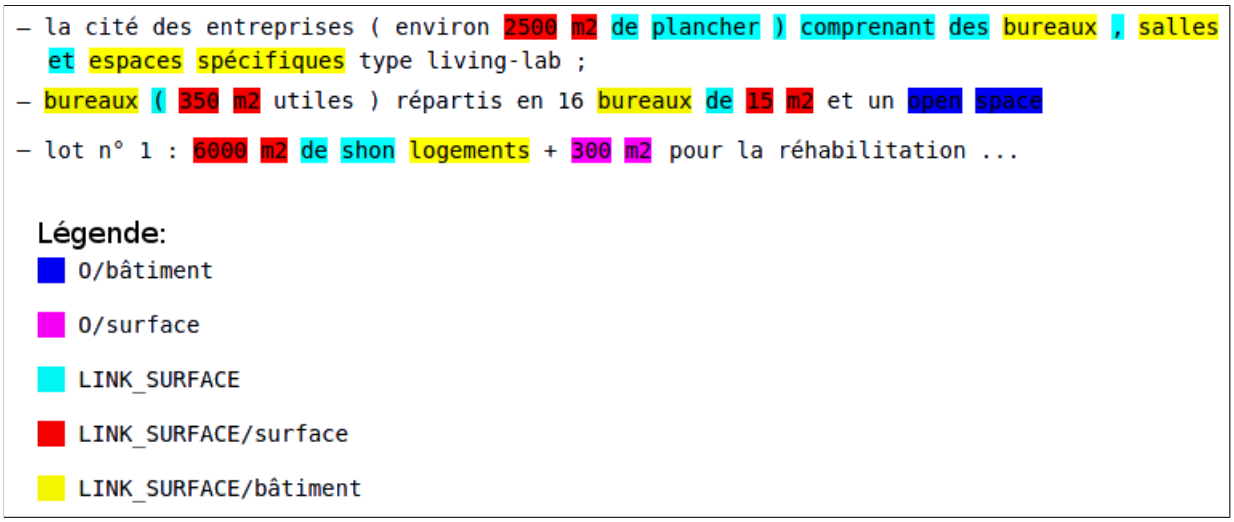

Figure 3.2 – Figure showing the labels used.

From the labels on the first sentence shown in the figure 3.2, we can easily infer the following relationship *"2 500 m2"* ***"is a surface of"*** *"bureaux + salles + espaces spécifiques"*. And therefore we just need to extract the sequence of elements (items) that have the following labels: **LIEN_SURFACE/bâtiment**, **LIEN_SURFACE/surface** or **LIEN_SURFACE**.

Features used

The features used to train our model can be split into two types: **Local features** related to one of the token of the sequence:

- **word.lower** : the uncased form of a token.
- **istitle** : Boolean feature that indicates whether or not the word starts with a capital letter.
- **lemma** : The word lemma.
- **POS** : "Part of speech" of the word.
- **type** : This feature can have one of the following values: **bâtiment** if the word is in a predefined set of buildings names, **surface**: A regular expression (regex) is applied to check if a token is a word (this feature turn out to be useful in the case of tokenization error, example: "200m2shon") or **0** otherwise.

The second type of features are **Long range features** that takes into account the context of tokens which reduces the semantic ambiguity of the words and have proven to be effective for sequence labeling tasks [91],[92]. Let x'_i be a feature set of local features x_i describing a token w_i after adding long range features, as described in the following equation: $x'_i = \langle x_{i-j}, ..., x_{i+j} \rangle$, where j is the context size. For example: for $j = 2$, $x'_4 = \langle x_2, x_3, x_4, x_5, x_6 \rangle$.

Used dataset

The data used for this experiment are taken from BOAMP [2] dataset, the electronic version of the French official journal (The French equivalent of the TED dataset). The BOAMP is used by DILA (Direction de l'Information Légale et Administrative) to publish public procurement notices.

The documents are indexed using (Lucene) search engine in order to select documents that are related to the construction of new buildings. The text is then split into sentences and then regular expressions are applied to detect building names or surfaces. We used (among others) the following Regex:

$((([0-9]\backslash s*)+((\backslash .|,)[0-9]+)*\backslash s*)(m\backslash s*2|m^2|ha(\backslash s+|\$)|hectare\backslash w*|metres\backslash s*((car)\backslash w*)*(lineaires*)*))$

2. http://www.boamp.fr

The sentences that contain a surface are segmented into tokens, then local and contextual (long range features) features are extracted. The initial dataset contains 2000 sequences. This sequences are labeled using a regex (the regex labeling is used as a benchmark), the result is then corrected by OctopusMind experts in order to have the final dataset that will be used in our experiments.

Experimentation and results

Several models for information and relation extraction are evaluated on this dataset. We compared different types of conditional random fields (CRF): with long range features and different context sizes, high order CRF (H-CRF) and semi-Markovian CRFs with. We have also evaluated a hidden markov model and a structured perceptron. We have used a Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS [93], [94]) method for CRFs weight optimisation. We have applied a *grid search* [95] algorithm in order to select the (l_1, l_2) coefficients for L-BFGS regularisation. Choosing the *grid search* method was mostly motivated by easiness of implementation and by the simplicity of the optimization problem, as we only have two variables to optimize. The number of iterations used to train each model has been set to 200.

For this experiment, we implemented a cross validation procedure with $k = 5$ folds (80% for train and 20% for test). The evaluation measures used for this experiment are: precision, recall, F1-measure and accuracy (The percentage of sequences which all labels are correctly predicted). The accuracy can be seen as the success rate on the sequence level compared to the other measures we use that are at label/token level. The results are shown in table 3.1.

From table 3.1, we can see that the CRF with long range features performs the best, with an accuracy of **76.04%.** This result confirms that the use of context aware features (long range features) increases the performance of the models. We also noticed that for this particular problem, having a context size bigger than 3 does not improve the result any further. The HMM got the worst result, as they are not suited for relationships extractions problems. We also note that by increasing the complexity of a model coupled with small size of the training data may cause over-fitting.

The table 3.2 shows the different measures evaluated for each label prediction taken separately.

	Precision	Recall	F1-measure	accuracy
CRF context (3)	0.932	0.932	0.932	76.04%
CRF context (2)	0.926	0.926	0.926	74.65%
Linear CRF	0.899	0.896	0.897	61.75%
CRF semi-Markoviens	0.897	0.899	0.897	67.74%
HCRF order(3)	0.878	0.878	0.877	66.36%
HCRF order(2)	0.884	0.882	0.882	63.59%
Structured perceptron	0.898	0.897	0.897	64.52%
Regex	0.884	0.851	0.855	66.89%
HMM	0.776	0.733	0.667	15.21%

Table 3.1 – Results obtained with the different models.

	Precision	Recall	F1-measure	support
O	0.954	0.943	0.948	3473
LINK_SURFACE	0.891	0.909	0.900	1764
LINK_SURFACE/bâtiment	0.931	0.940	0.935	315
O/bâtiment	0.807	0.800	0.803	115
LINK_SURFACE/surface	0.973	0.973	0.973	524
O/surface	0.767	0.793	0.780	58

Table 3.2 – Scores obtained by CRF context (3) model for the different labels.

By further analysing the results, in particular while checking a few miss labeled sentences, we have noticed that some of the errors were due to the human expert errors (an example is shown in figure 3.3). The number of errors in manual annotations is estimated at 1%. In those cases the model gave better results than the ground truth, which proves a good generalisation capability. An example of a miss-labeled training sequence is shown in figure 3.3

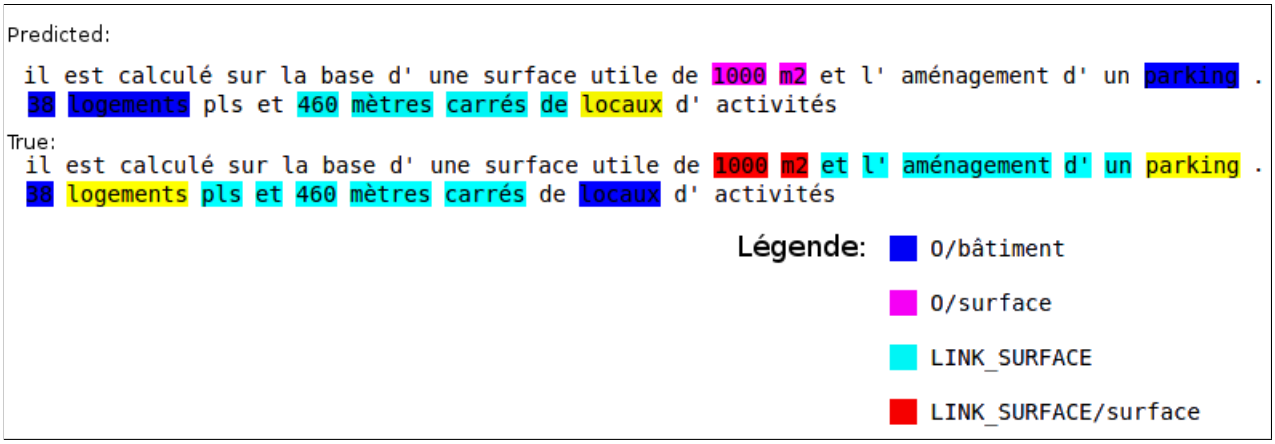

Figure 3.3 – Figure showing error in sequences labeled by a human expert.

Also by analysing the CRF weights after training, we notice about the variable **type** that, while it helps for classifying surface and buildings concepts, the model does not rely solely on this variable for prediction: as can be seen from the following table.

Feature	Label	Weight
lemma:m2	LINK_SURFACE/surface	1.087
type:surface	LINK_SURFACE/surface	2.296
type:bâtiment	LINK_SURFACE/bâtiment	4.674
type:surface	O/surface	1.452
word.lower():m2	LINK_SURFACE/surface	1.079
POS:NOUN	LINK_SURFACE/bâtiment	1.561
type:bâtiment	O/bâtiment	3.551
-1:type:surface	LINK_SURFACE/surface	0.919
-1:lemma:mètre	O	-0.53
POS:NUM	O	-0.69
lemma:local	O	-0.81

Table 3.3 – Summary of weights associated with the most discriminant characteristics, by label, for the CRF context (3).

3.2.3 Renewal information extraction

We also tackled another task similar to the surface detection problem: tender renewal detection. Early detection of interesting leads is a priority for companies, in order to anticipate their strategy and assess business opportunities. Thanks to these weak signals[96], companies can plan their actions accordingly. Information on the tenders renewal give them a strong competitive advantage.

Methodology

The "tenders renewal" problem can be considered as a named entity detection problem, as we need to extract a set of pieces of information that describe the project renewal, namely:

- "période": The time limit for a call for tender.
- "periodicité" (periodicity): Describes the number of renewals the call for tender may have.
- "début du marché": The starting date of the project described in the call for tender.
- "fin du marché": The ending date of the project described in the call for tender.
- "début de la reconduction": The starting date for the renewal.
- "fin de la reconduction": The ending date for the renewal.
- "délai du marché": The time allocated for the project described in the call for tender.
- "délai de la reconduction": The time allocated for the renewals (i.e. a call for tenders that has a "periodicity" of 3 and a "période" of 1 year the value of this label should be "3 years").

The figure 3.4, illustrate some example of labels used.

Used dataset

Due to the lack of a suitable dataset to train a model for this labeling task, it was necessary to create a training database manually. To do so we used Lucene search engine (through the Elasticsearch encapsulation) queried on OctopusMind's database to select all documents that may contain the word "reconduction" [3] (french word for

3. We also note that we only select french documents.

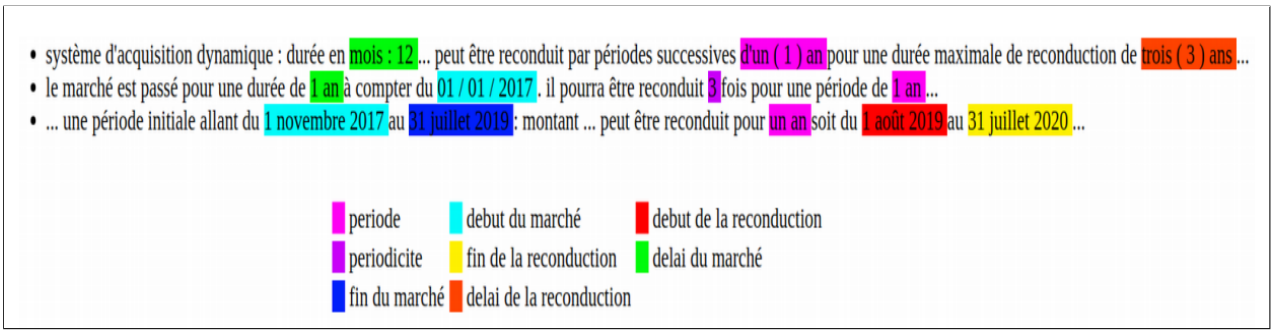

Figure 3.4 – Figure showing the labels used for tender renewal detection.

"renewal") and its different variations of occurrence. Then we apply a filter to select a 10 documents from 50 different sources (to take documents with different structures and syntaxes), we end up with around 500 documents. This dataset can be considered to be sufficiently large and diversified to ensure a good training set for our CRF model.

We developed a homemade tool (as shown in figure 3.5) to allow experts to efficiently label this dataset.

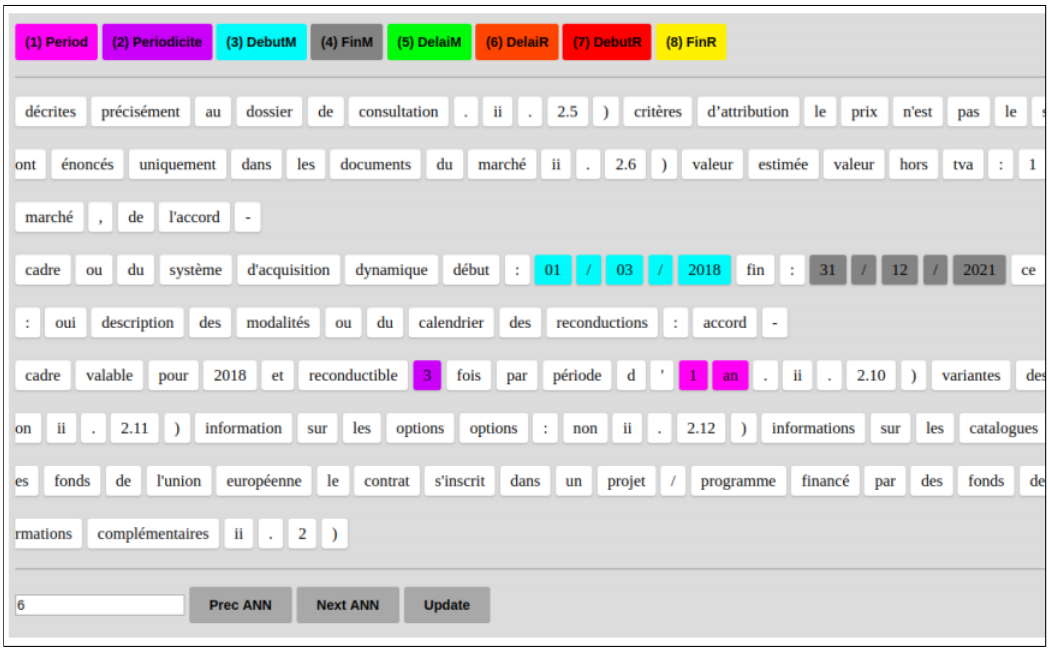

Figure 3.5 – Figure showing the tool developed for sequence labeling.

Features used

For this task we used similar features to those used for the surface extraction problem. These features can be split into two types:

Local features related to one of the token of the sequence:

— **word.lower** : the uncased form of a token.

— **istitle** : Boolean feature indicates whether or not the word starts with a capital letter.

— **lemma** : The word lemma.

— **POS** : "Part of speech" of the word.

— **isdigit** : Boolean feature that indicates whether or not a token is a number (defined using a Regex).

As for the surface extraction task, we also use **Long range features**.

Experimentation and results

For this sequence labeling task we used the CRF context (3) model. The following table 3.4 shows the results obtained by applying a cross validation (k=4)

	Precision	Recall	F1-measure	support
period	0.84	0.92	0.87	700
debutr	0.13	0.43	0.20	7
delaim	0.89	0.88	0.89	1173
delair	0.78	0.85	0.82	96
dureem	0.92	0.93	0.92	634
finm	0.80	0.83	0.81	303
finr	0.60	0.85	0.70	116
periodicite	0.88	0.97	0.92	454
avg/total	0.87	0.90	0.88	3483

Table 3.4 – Scores obtained by the CRF context (3) model for the different labels.

From the table 3.4 we can notice that the CRF context (3) model seems to perform well for most of the labels with an F1-measure greater than 82%. The only problem is with the label "debutr" for which the precision is very low (13%): this is caused by the lack of training examples in the dataset (only 7 sentences of 4000 contains this label).

By adding more example of this label, the performance for this specific label should significantly increases.

3.2.4 Financial data extraction

The costs of projects in call for tenders and contract award notices, is a very important information for OctopusMind's clients. Indeed, the information that describes the financial value of a project (the contract award value) allows companies to better position themselves, by knowing what share of market their competitors control and the price a customer is able to pay (market analysis) for a given project.

Some financial data contained in call for tenders documents are requirements that can be disqualifying for the bidder, such as the presence of a minimum turnover required for the candidate. In a somewhat more indirect way, some companies use the information from the contract budget estimate or the amount awarded for a contract to contact the contract winner in order to sell them services necessary for project completion.

Therefore extracting such information is strategic.

Methodology

The financial data extraction task can be seen as a named entity recognition task. As we want to be able to identify the value of the contract and extract the allocated fund of each lot [4] in the award notice, and provisional budgets in the case of the call for tenders.

The problem can be decomposed into 3 steps:

— The lot detection.

— Budget extraction.

— Budget classification.

First we need to detect in a document the text describing each lot (we note that the description of a lot can be located in different part of the document). The second step is related to the amount detection (detecting a "money amount") from the text. And finally, the third step consists in classifying the detected "money amount" into one of the following categories:

4. Some project in call for tenders are split into parts (lots) each lot will have a different allocated budgets

— The "Minimum" amount that can be allocated.
— The "Maximum" amount that can be allocated.
— The "Total" cost of a project.
— The minimum "Turnover" a company needs to have in order to have its offer considered.
— The "N/A" label means that the money amount will not be considered as a budget (usually administrative fees)
— The "Firm" section which is only applied to project that are achieved in several steps, planned at different moments (Usually big construction project), the firm section will be the budget of the section that must be carried out.
— The "Conditional" section is a budget that is only applied if the company respects a set of conditions listed in the call for tenders document.
— "Bonus" amount is only applied for architects and defines/sets the amount architects will be payed when submitting their proposal even in the case of rejecting their proposition.
— The "Travaux" label is used for construction cost estimates.

For the lot part detection, we opted for the use of a Regex coupled to a simple heuristic. The regex used is shown below and the result of this regex is shown in figure 3.6:

```
(?:[\b\s]N°|lot)\s*(?:n|\(s|\s)[^\d+]{0,3}\s*(?P<num_lot>\d+)
```

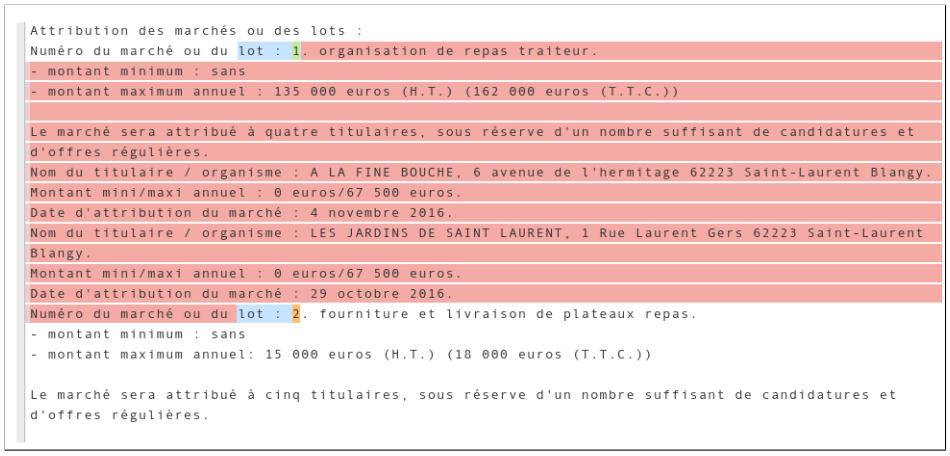

Figure 3.6 – Figure showing the result of applying the regex for lot detection.

In order to regroup different parts of text that describe the same lot, we rely on the lot id (usually a number or a code). The paragraphs that have the same lot id are concatenated.

For the second and third part of the problem (detecting and classifying a "money amount") we decided to solve jointly the detection and the classification using a CRF (we use the CRF context(3) used in 3.2.2).

dataset

For the purpose of this task, we have built a dataset containing around 4000 sentences manually annotated by the by experts. The sentences where selected using (Lucene) search engine in order to select documents that contain a currency (Euro,€,$,etc.). The documents are then split into sentences and the budget token is detected using a Regex. The sentences that contains budget are the annotated.

The figure 3.7 illustrates the interface used to label the dataset.

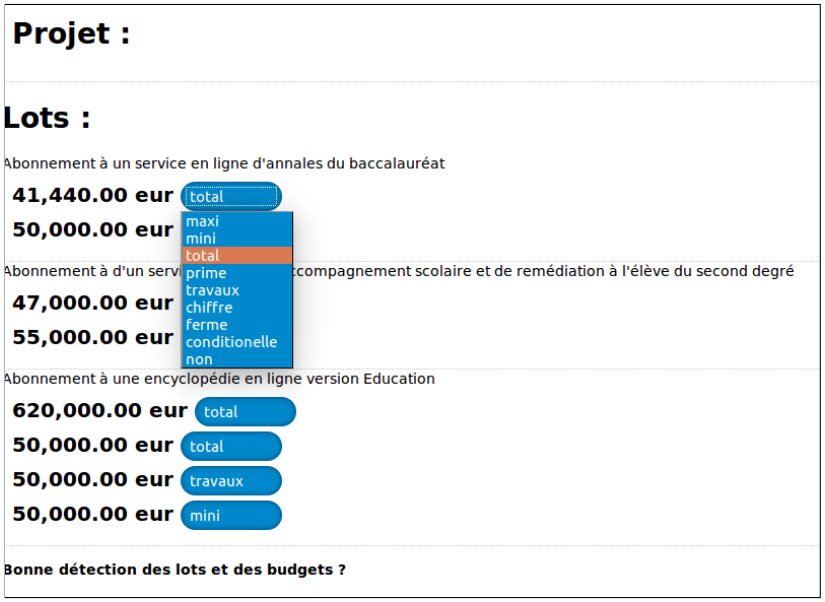

Figure 3.7 – Figure showing the validation interface of the detected amounts.

Experimentation and results

The results given by the CRF context(3) model are shown in the table below using a stratified (the distribution of the categories in the train/test sets is preserved according to the statistics evaluated on the whole corpus) cross validation (with k=5 folds).

	Precision	Recall	F1-measure	support
Turnover	0.00	0.00	0.00	16
Conditional	0.00	0.00	0.00	13
Firm	0.00	0.00	0.00	4
Maximum	0.70	0.76	0.73	542
Minimum	0.73	0.70	0.72	382
N/A	0.72	0.51	0.60	284
Bonus	0.00	0.00	0.00	4
Total	0.85	0.86	0.86	874
Travaux	0.67	0.44	0.53	54
avg/total	0.76	0.74	0.74	2173

Table 3.5 – Scores obtained by CRF context (3) model for the different labels.

As we can see from table 3.5, the model is not able to correctly handle unbalanced datasets. Therefore it cannot predict at all classes with small number of representation.

The results were not acceptable according to OctopusMind's criteria. So we used another approach that revealed to be more efficient (and faster compared to the CRF). This approach relies on splitting in two sub-problems the detection and the classification of the budget as was originally planned. For the detection we decided to use a regex as the problem is not too complex. An example of a used regex is shown below:

```
(((?:[0-9](?:\s|\.|,)*)+)\s*(M)*(EUR\w*|€|eur\w*|chf))
```

For the classification part, we developed a context aware version of bag of word. In our case the context is defined as the word surrounding a budget (any money amount). The method consists in segmenting the sentence in which the budget appears into two parts and applying a Bag Of Word representation by considering the words to the left of the amount and those to the right as different features, in such a way that it is possible to distinguish the words occurring in each part. As shown in the example below:

"Valeur totale du marché (hors TVA) :Valeur 92 915,71 euros ou Offre la plus basse"

The sentence will be represented as follow:

```
['left_valeur', 'left_totale', 'left_du', 'left_marché',
'left_(','left_hors', 'left_tva', 'left_)', 'left_:',
'__budget__', 'right_ou','right_offre', 'right_la',
'right_plus', 'right_basse']
```

Then we use a SGD classifier based on this representation in order to predict the budget category.

The table 3.6 shows the results obtained by this method using a stratified cross validation (k=5).

	Precision	Recall	F1-measure	support
Turnover	1.00	0.75	0.86	16
Conditional	0.40	0.31	0.35	13
Firm	0.67	0.50	0.57	4
Maximum	0.87	0.88	0.88	542
Minimum	0.88	0.86	0.87	382
N/A	0.75	0.83	0.79	284
Bonus	0.67	0.50	0.57	4
Total	0.94	0.93	0.94	874
Travaux	0.88	0.78	0.82	54
avg/total	0.88	0.88	0.88	2173

Table 3.6 – Scores obtained by an SGD model using the left/right BOW for the different labels.

We notice from the table 3.6 that our last method outperforms the CRF model and is even able to predict classes with very small number of observations as shown in tables [3.6,3.5].

This method is also more efficient in term of required computation time (the preprocessing time is not taken into account for this experiment) as illustrated in the table 3.7. We note that the implementation of the CRF is done using Crfsuite [5] [97], scikit-learn [6] [98] for The SGD.

5. http://www.chokkan.org/software/crfsuite/
6. https://scikit-learn.org

	Training	prediction
CRF context(3)	$11s$	$1.15s$
SGD (left/right)	$21ms$	$4ms$

Table 3.7 – Time needed by the SGD and CRF for training and for predicting the whole dataset (2000 Sentences)

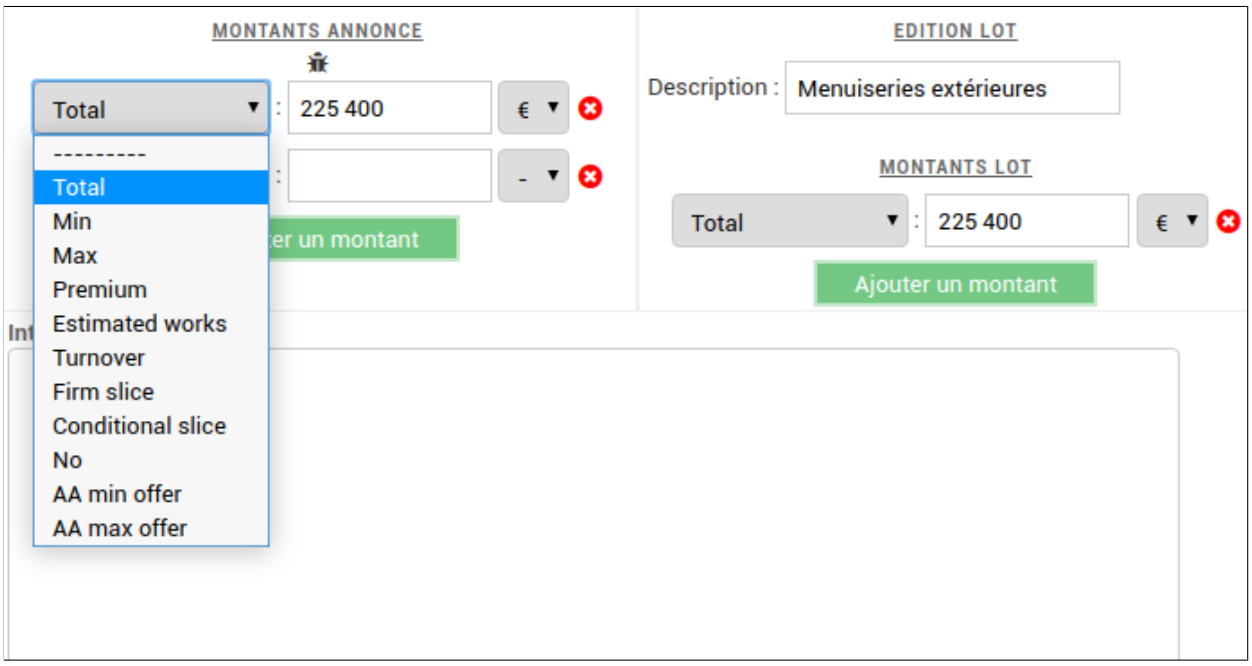

Figure 3.8 – Figure showing the validation interface of the detected amounts in production environment.

The final model is now integrated into OctopusMinds software environment, and used in a daily basis. The economic information extracted by our model are stored with the objective to sell it to OctopusMind's clients or to be used by the company experts (we also note that this information is indexed and can be used as a filtering criteria by a search engine).

OctopusMinds review the result of our model in the production environment and modify their predictions if needed by using the interface shown in figure 3.8 (allowing them to add/edit amounts and changing the classes that have been incorrectly predicted).

The corrections made by the experts are used to periodically re-train the models.

3.3 Recommender system

With the continuously increasing growth rate of documents managed by OctopusMind, the creation of a tool to easily access the most relevant information for the platform's users is a strong requirement. In addition, as a strategic and economic intelligence service company, the relevance of the results that are produces is a primary factor for OctopusMind.

In order to develop such a tool we have decided to analyse users interactions with the platform and especially the search engine usage (the platform offers a search engine that allows for filtering the results in very detailed ways using different tags and logical expressions). We noticed that around 95% of the users limit themselves to basic queries using only few key-words. By that we understand that the majority of the users will make minimum effort while still expecting highly relevant results.

The objective is then to develop a recommender system that allows the user to access the most valuable information (the relevant documents according to their needs (activity field)) that suits his needs. And this have to be done in a way that minimizes the user's interaction effort.

The most widely used method for item recommendation is collaborative filtering. It consists on making automatic predictions on a user interests based on preferences and tastes information of other users.

The **collaborative filtering** approach is based on the assumption that if a person A has the same taste than a person B regarding a topic, then the person A is more likely to have the same taste than the person B on different topic. For example, a recommendation system based on collaborative filtering applied on an online bookstore could make predictions about which book or author a user is likely to buy given a list of users tastes or buying history [99].

The **collaborative filtering** methods are not suitable for the kind of data we are dealing with due to the nature of public procurement documents and especially for the following reasons:

— The documents have a limited time span, as each project have a deadline: after the deadline the document is no more relevant, thus cannot be recommended to a user.

— The documents need to be recommended as soon as they are published on the platform, which means that the documents must be recommended before

gathering any information about user interests concerning this document.

Another common approach is **content-based filtering** [100], which is done by assigning a description (features [7]) for each item we are recommending and for the users profiles. Then recommending an item is done by comparing its description to a user profile description. This comparison is achieved by using a classifier that will learn user tastes based on items description.

The recommendation system [8] used by OctopusMind is close to a **content-based filtering** model. It was built on top of a search engine (elasticsearch [9]).

This was implemented as follows:

— The features of an item (document) are the words contained in the text and also a set of custom keywords ("thésaurus" [10]) assigned by the experts for each reviewed document.

— For each SEPAO [11] user, an elasticsearch dedicated query is built that contains a set of keywords describing the user's interest and a list of "thésaurus" [12] terms that may interest him. The query results are then manually reviewed by the expert team before sending them to the users. Also no recommendation system was established for other types of users (Non premium).

This approach is heavily reliant on human expert's annotations. In order to improve this approach we focuses on three aspects:

— Reducing the human experts workload.

— Broadening the recommendation scope to all users (Not limiting to premium users).

— Offering document based recommendation.

3.3.1 Thésaurus classification

The automation of "thésaurus" based classification can significantly reduce human experts work load. In order to improve the previous recommnender system we decided

7. The features can be any way to describe an item, in our case we use keywords
8. Was implemented in a simple form before the start of this thesis
9. https://www.elastic.co/fr/
10. Note that this keywords are not necessarily explicitly present in the document, but usually guessed from the understanding of the text, the "thésaurus" categories have distinct semantic groups, for example, some may characterize the call for tender type, others the sector, others the necessary skills, etc.
11. Users with premium subscriptions
12. "Thésaurus" are a set categories crated by OctopusMind for internal use, allowing to classify the documents in order to route them toward the appropriate users.

to start by addressing this objective.

This "thésaurus" classification based problem can be seen as a multi-label classification, consisting in classifying each document into different "thésaurus" entries.

We started by extracting the already annotated documents, ending up with a dataset containing around 60K documents (356 classes). We filter some of the "thésaurus" classes that require a more in depth text understanding and that could be too complex to predict using a classifier that is only trained on raw plain text.

The following example illustrates some of the classes we have filtered out:

— "Marche_20M_plus": a class concerning projects that will cost more than 20 millions euros.

— "C2_Couv_region_depart_plus_150000hab": a class related to a project executed in a department or state with more than 150K inhabitant.

— "A_Hopital_EHPAD": a class concerning projects where the buyer is a health institution.

We note that the examples given above can be challenging to categorize if the classifier is only trained on textual data. For example:
"C2_Couv_region_depart_plus_150000hab" will need external information (number of inhabitant in a region) and "Marche_20M_plus" need the budget detection and conversion to numerical value to be able to correctly predict its thésaurus class. In the other hand, this kind of problems can be resolved easily using simple expert rules.

This classification problem was first tackled before the work we have done on document embedding (before developing LSA+W2V method). So we used as document embedding method a weighted sum of Word2vec vectors. The weighting is done by separately averaging representations of words belonging to the same semantic cluster (concept). The resulting vectors associated to a document are then averaged.

Lets take for example the sentence:

```
"Création d'une application mobile pour ios et android"
```

After tokenization, bigram [13] extraction and stop words removal, we end up with the following tokens list:

```
['creation',
'application\_mobile',
'ios',
```

13. The bigram $application\_mobile$ is considered as one token.

```
'android']
```

The bigrams are formed according the method used in [3], based on the unigram and bigram counts:

$$score(w_i, w-j) = \frac{count(w_i w_j) - \delta}{count(w_i) + count(w_j)} \tag{3.1}$$

Where w_i, w_j are consecutive words and δ is used to prevents bigrams with infrequent words to be formed. The bigrams $w_i w_j$ with a score above a chosen threshold are then used as bigrams and considered as tokens during tokenization.

After calculating word2vec representations for these words, we use k-means clustering algorithm in order to regroup the similar words into clusters (5000 clusters have been constructed). Then these clusters are reviewed by experts and the clusters containing mostly stopwords [14] have been removed.

The worlds from the same cluster are then grouped together.

```
[ ('creaton', Cluster_6),
  ('applicaton_mobile', Cluster_1242),
  (('ios','android'), Cluster_1519)]
```

Finally, the words belonging to the same cluster are averaged together, and the resulting vectors describing a document are then summed up as shown in the example below:

```
V(sentence) = W2V(creation) + W2V(application_mobile)
+ 1/2(W2V(ios) + W2V(android))
```

This method is applied on the title and the lots description of a document. The resulting vector is then used as input to a MLP (Multi-Layer Perceptron) classifier with 2 hidden layers.

The results obtained with this initial model are shown in table 3.8 (the results obtained with lsa+w2v are given also for comparison).

The model used in production by now is based on lsa+w2v (cf. 2.3.1). The LSA is created using tf-idf bag of word vectors and the text used to generate the tf-idf is based on the full description of the documents once the filtering out of the administrative information (using the same method as in 1.2.2) has been performed. The model has

14. Stopwords specific to the public procurement vocabulary.

	Precision	Recall	F1-score
w2v	0.71	0.48	0.54
LSA+w2v	0.75	0.53	0.59

Table 3.8 – Scores obtained on the initial dataset used for the "thésaurus" classification task using a train/test split of 80%/20%.

been re-trained using additional data (2 years of additional data). The results in the table 3.9, shows the performance of the different models after two years of periodical training (40K more documents)

	Precision	Recall	F1-score
w2v	0.72	0.58	0.63
LSA+w2v	0.73	0.62	0.66
CnnHAtt	0.72	0.69	0.69

Table 3.9 – Scores obtained with the enriched dataset, used for the "thésaurus" classification task using a split train/test of 80%/20%.

We notice from these result tables [3.8,3.9] that adding more data improves significantly the performance. This is because the initial data and also the classes are largely unbalanced. The following table shows some statistics on the two datasets. (initial and enriched)

	Average document per class	Median documents per class
Initial dataset	622	180
Final dataset	1058	4653

Table 3.10 – Comparison between the initial and the enriched datasets for "thésaurus" classification.

The result can be tweaked by increasing/decreasing the MLP threshold in order to favor the precision or the recall. In our case the model is used as a tool for decision-making support by helping the experts to label document in a more efficient way. Hence, we focused mostly on the recall, as it is easier to delete irrelevant labels.

The precision recall curve is shown in the figure 3.9.

We got good feedback from the experts who use this tool. Allowing them to accelerate significantly the labeling of documents (the gain is about 25% compared to the

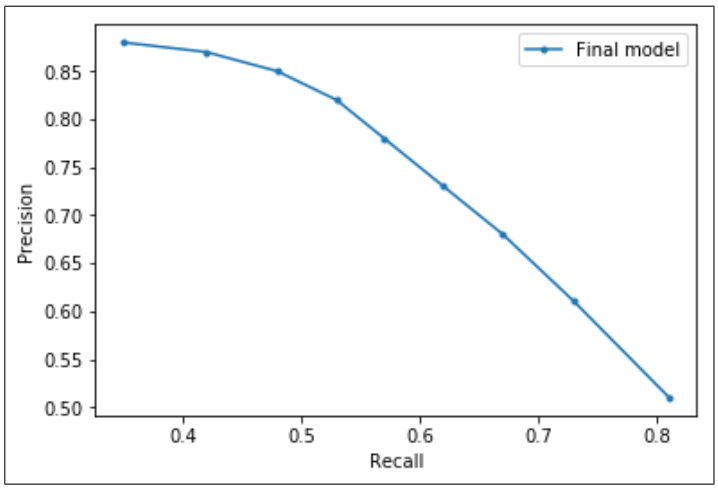

Figure 3.9 – Precision recall curve for the model used in production.

previous manual tagging procedure). We can also notice that the amount of documents per class is relatively low for neural networks training. We can conclude from table 3.10 that the more training data is available, the best is the model. So the performance of the model is expected to periodically improve with the increase of available labeled data with time.

3.3.2 Activity classification

The second aspect to improve is to broaden the recommendation service to all users. This means allowing non premium users to have easily access to the call for tenders that may interest them and recommending them the most relevant projects according to their trade/profession/"field of activity" with minimum effort.

The approach we decided to follow is to label each document according to its field of activity (trades). In addition to that, each user will be requested to fill a list of trades relative to their profession. By achieving this, the users can access recommendation directly after creating an account and selecting the area fields that are relevant to his profession. This is achieved through the interface presented in figure 3.10.

The first task was to create a list of labels (trades). This was achieved based on the CPV (Common Procurement Vocabulary) from the TED dataset[7].

The CPV codes were not used directly but combined in a way to create a number of classes that cover all trades while being sufficiently represented. The CPV codes were analysed by the expert team and we end up with 59 classes. For example the CPV

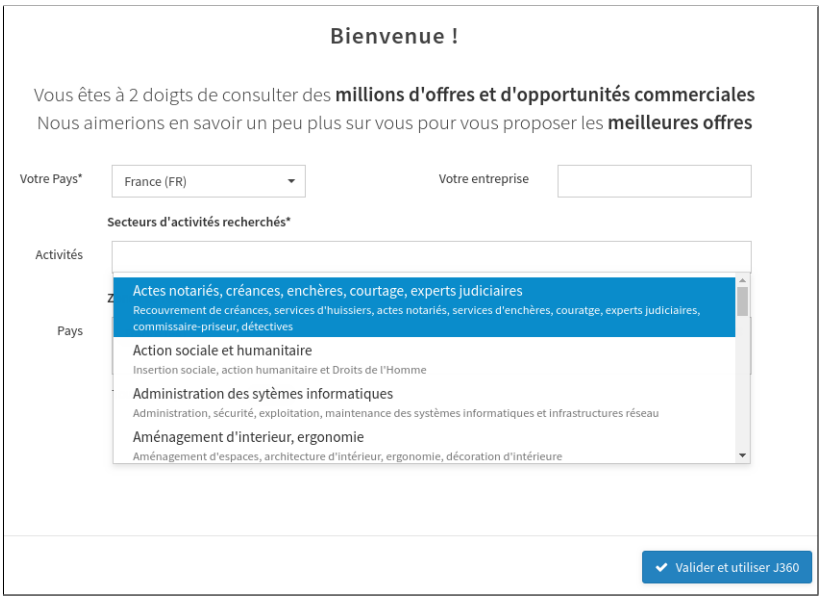

Figure 3.10 – J360 web application subscription interface.

codes for "Maple sugar and maple syrup" and "animal product" will be in the same class "food and agricultural products".

We then train an MLP with 2 hidden layers (500 neurons per layers) on the TED dataset with the transformed classes. The vectorisation method used is LSA+W2V. The Word2vec vector is created in the same way as for "thésaurus" classification task and the LSA vector is created using the filtered version of the full description (after filtering out the administrative information).

The resulting vector is then used as input to the MLP.

The results obtained with the initial model are shown in table 3.11.

	Precision	Recall	F1-score
SGD(tf-idf)	0.74	0.36	0.43
W2V	0.72	0.56	0.62
LSA+W2V	0.83	0.78	0.80

Table 3.11 – Scores obtained on the initial dataset used in "thésaurus classification task using a split train/test of 80%/20%".

This model has been deployed into the production environment and is periodically retrained.

We have then decided to use the activity tags as keywords on the search engine interface as shown in figure 3.11, so the users can use them to search projects that are not directly related to their activity (for example an electrician is usually interested in a construction project).

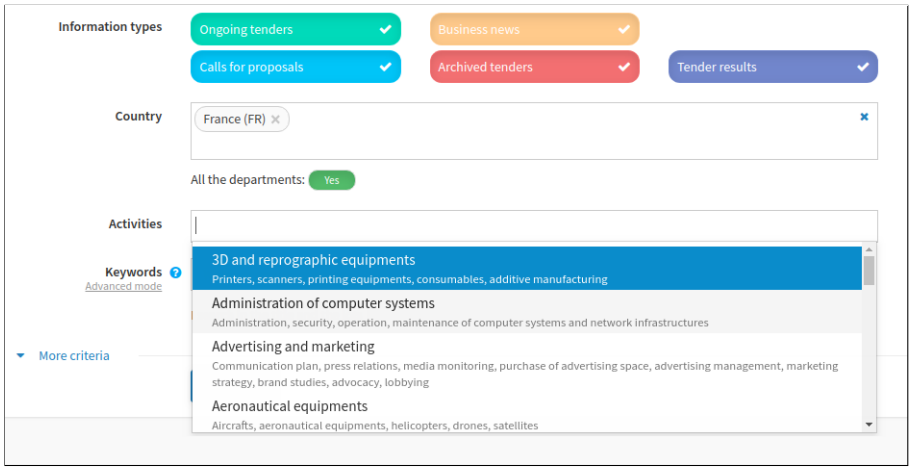

Figure 3.11 – Figure showing the query creation interface.

3.3.3 Document based recommendation

The third improvement aspect is establishing a recommendation system based on documents. Allowing the users to access a set of similar documents to the one he is currently reading.

A user experiencing the web application from external sources (a user that is not a subscribing user yet) would be able to obtain recommendation and access similar content only based on the text of a document. The choice of only considering the text while doing document recommendation is motivated by the fact that this way of exploring the database (jumping from a document to another) can allow the users to access documents that may be related to their field of activity that they couldn't see otherwise.

Let's assume this situation where a user who is interested in selling web marketing services is connected to the platform searching for projects in his field. He can search based on the activity classes ("Websites and mobile apps" class in this case) or by typing keywords on the search engine and by using one or both of these methods. The returned results can be as follow:

— Using only activity classes the projects that will be recommended will cover a broad range of professions within the activity field including websites and app creation, web hosting, etc. Hence, more filtering is needed to access the most relevant information.
— Using the search engine (with or without selection of activity classes) will only return the projects that exactly contains the words typed in query and not take into account synonyms for example.

Recommending documents based on semantic similarity can mitigate these two issues. In order to achieve this we need to be able to accurately calculate a semantic similarity between documents.

For the initial model we have used the weighted Word2vec sum method (the same method used for the "thésaurus" classification task and activity prediction) and we used cosine similarity to estimate document similarity. To evaluate the performance of this approach we decided to use it jointly with a kmeans (k=1000) algorithm. We applied this clustering on all the french documents contained in our database (which represents about 3.5M documents by the time the experiment was done).

Manually checking all clusters was too costly, so for the evaluation we asked the experts to randomly pick up 500 clusters and check if the 10 closest documents to the centroid of the cluster are coherent.

The following example illustrates a cluster:

```
- Séjour APPN Côté d'Opale (ou bretonne ou normande)
- SORTIE A PARIS
- Voyage pédagogique au Mémorial de Caen et Plages du débarquement
- Sortie site archéologique Chateaubleau
- Sorties Pédagogiques COLLEGE MAPA 12020C
...
```

We got promising feedback from the experts regarding the quality of the obtained clusters. However in a cluster, the farther a document is from the centroid the less relevant it is expected to be.

Then we created a small dataset containing 2000 documents (here a document is only represented by its title) randomly selected from different clusters. for each of this document we get the 10 nearest neighbors documents using our method. The experts then validate each results by answering the question "is this neighbor similar to the

query document?". The notion of similarity for the expert is based on profession concept (the neighbors are similar to the query if they are related to the same profession).

The accuracy/precision that we obtained on this dataset is 63%. This score is moderately satisfactory, but the model was still deployed into the production system of the company, since the experts find the result acceptable for a first model and clustering results seemed globally coherent. Also one of the objective of this model is to allow the users to explore the data in a progressive way (going from a profession to another progressively).

In the production environment, the search for the k documents that are the closest to a given document (query document) is done with the K-nn (k-nearest neighbors) method using the cosine similarity. It consists in calculating the similarity between the document query and all the documents in the dataset. This can be computationally challenging when working with large datasets.

In order to overcome this issue we use k-means (k = 10000) clustering algorithm, to first split the dataset, into clusters that regroups similar documents. We first apply the k-nn search on the clusters centroids to select the closest clusters. Then we apply a second K-nn on the documents belonging to the selected cluster(s). The returned documents are then used as recommendation.

We note that it is preferable to use multiple clusters for the second K-nn to avoid omitting some documents. For example, lets assume that the documents D_a and D_b belong respectively to the clusters with the centroids C_a and C_b, and suppose that the following properties are verified:

— $cos(D, D_b) < cos(D, C_a)$
— $cos(D, C_b) > cos(D, C_a)$
— $cos(D, D_b) < cos(D, D_a)$

Where cos is a cosine distance. In this example, even if D_b is closer to D than D_a it will not be returned in the recommendation. This method allows us to get a huge time gain. For instance a k-nn on a 2M documents dataset will calculate 2M cosine similarities and with this methods (assuming we have 1K clusters and each cluster contains 2K documents) if we consider two clusters, we only calculates 5K distances.

The following figure 3.12 shows the interface used to list the documents that are recommended.

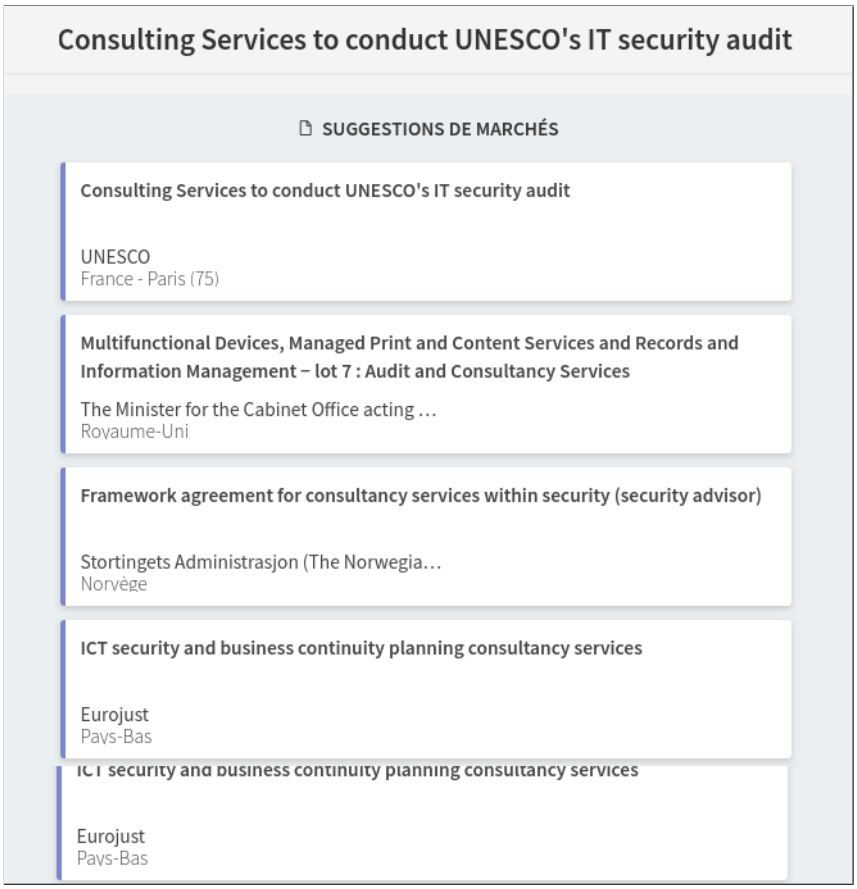

Figure 3.12 – Figure showing the recommendation interface.

3.4 Conclusion and perspectives

In this chapter we focused on the different solutions we developed with the aim of including them into the OctopusMind software environment, for daily production uses. We have presented some of the information extraction tasks (financial data, surfaces) allowing the company to achieve an important technological leap, essentially by offering the users to access relevant information that are manually impossible to get on big datasets.

In other hand, we also developed a recommendation system that minimises the human efforts for developing this relevant information access service. The recommender system is evaluated on a small dataset. Even if the experts seems to be pleased with the results we need to develop a consolidated evaluation workflow.

We also need to evaluate our newly developed document embedding model (CnHAtt) on similarity tasks and using it as the default model for classification tasks.

As a perspective, we will need to update our different models to work with multiple languages. For the time being our different NLP models only work with French and English languages and this is achieved while creating a model for each languages.

In addition to that we are aiming to explore the unsupervised approaches for relationships extraction in an unsupervised fashion [101],which will allows us to discover new relationships that we may not notice acknowledge otherwise, we also note that unsupervised relationships extraction is less costly compared to the supervised way, as it does not need labeled data.

Also we need to further develop the interactions with our different modules. One line of on-going research is to create new information at a higher level of analysis, mostly by aggregating already existing one.

An example of information extraction we started working on is a market analysis task. By trying to analyse financial information and producing statistics on a specific industry. The first attempt we made, is to combine the financial data extraction model we have developed with the document based recommender system. This allows to get more elaborated financial information about a certain type of project.

For example by selecting a document we are able to list the k most similar projects (we also use a similarity threshold to reduce the noise). And then we are able to calculate the average, minimum and maximum cost for this type of projects.

Also listing the bids winners of similar projects with the project costs, allows the

users to identify their competitors and their prices. This information is important to better position OctopusMind clients into their market.

The preliminary interface for accessing this tool is given in figure 3.13.

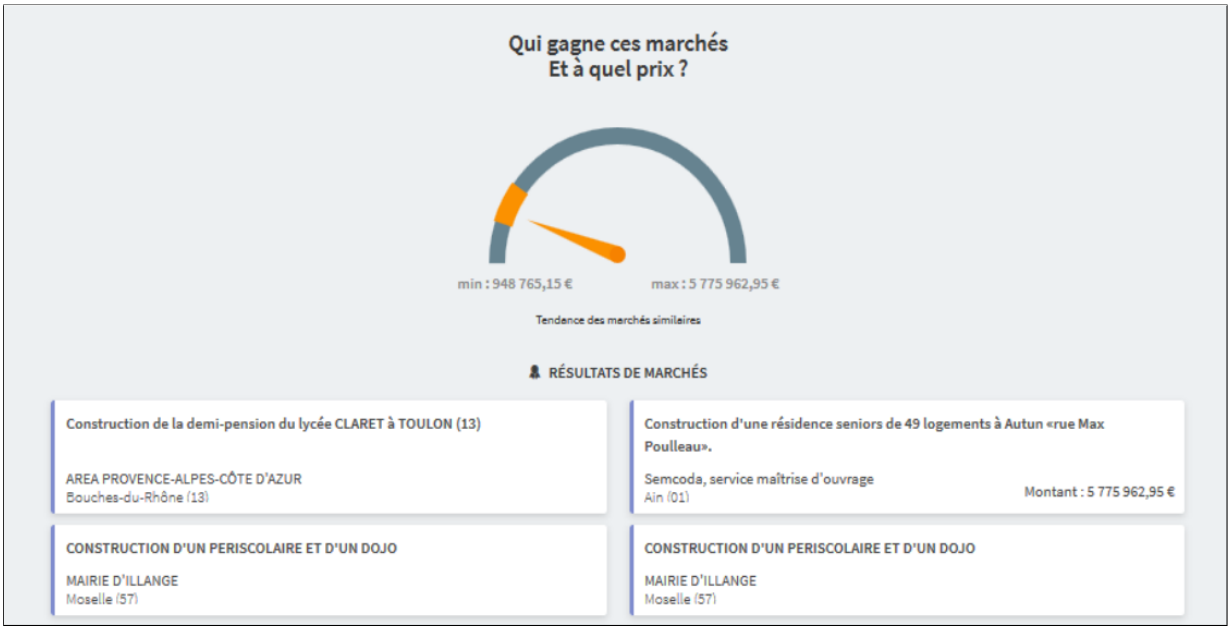

Figure 3.13 – Figure showing the market analysis tool on j360 web application.

CONCLUSION

Users of online recommendation platforms have high expectations regarding information quality obtained with little effort. Any system which aims at providing relevant answers to such platform users must take this fact into account.

This is especially challenging when working with unstructured documents. This is the case at OctopusMind, whose major service is to recommend public procurement related documents (essentially call for tenders). Before routing the documents to relevant users, each document has to be analyzed, so that important information can be extracted (such as: financial data, execution location, etc.) and eventually categorized into a set of classes.

Our thesis work was focused at providing OctopusMind with a set of tools that allows this process to be automated. Based on NLP methods, the final product aims to be included in the already existing software environment.

To achieve such a goal, we had to tackle a broad range of NLP tasks including named entity recognition, relationship extraction, document classification and document embedding. Keeping in mind that OctopusMind is an SME, it comes with its set of challenges, as all the solutions we provide have to be efficient (with low memory usage and execution time) to cope with the hardware limitation.

This thesis has been structured into three chapters, which reflects the different NLP problems we have to tackle. We also note that the order of the tasks described in the different chapters does not reflect the orders in which we dealt with them. In most cases the different tasks have been addressed according to the company strategic planing.

The need to create a corpus related to the public procurement domain was of a high priority, in order to train or evaluate any machine learning approaches and solving the automatizing problems that Octopus Mind had planned. Chapter 1.1 describes the process we go through to generate a multilingual corpus composed with documents published by the European union and related to public projects covering nearly all the business activity.

The corpus is divided into two sub-datasets, fd-TED and par-TED. The fd-TED corpus that we have used at several occasions during this thesis has been created with

the goal of multilingual text classification which suits the international market targeted by OctopusMind. The par-TED corpus consists of a set of translated sentences. It was mostly a contribution toward the NLP community, especially in the scope of the machine translation. Many of the problems raised by OctopusMind can be considered as document classification tasks, and when it comes to text classification, the embeddings methods we have tested and developed show significant impact on the quality of the provided service.

Chapter 2 describes our contributions in this area. We started the chapter with a survey of different document embedding methods, but due to the nature of the public procurement document and the hardware limitation we are held to, we have developed our own methods based on the analysis of the state of the art techniques.

We started by combining LSA and Word2Vec averaging methods as they bring complementary views on lexical semantics, providing a local and global semantic overview. The results were acceptable but still lacked proper term weighting, in the sense that all the words in a document will contribute equally, this is especially the case for Word2vec averaging. The negative impact of this uniform kind of weighting is even more significant due to the noisy nature of public procurement documents. Attention mechanism was a natural step forward to cope with this issue.

CnHAtt, our second contribution in the area of document vectorization, was developed using CNNs as an efficient (time wise) method to create document embedding by only taking into consideration the most relevant words according to the classification task at hand. The hierarchical structure of the CnHAtt, allowed legal and administrative sentences to be discarded by giving them lower weights. The results are promising, but more experiments need to be done to finalize the different possible variants of this method.

Chapter 3 was essentially centered on the application benefits of this thesis. The result of the work described in this chapter are current solutions to real world problems. As stated above, the finality of this thesis is to help OctopusMind establishing an automatized process for on-line public market monitoring. Information extraction was of a high priority, as it allows us to automatically gather business intelligence features. We have established a well-functioning, reusable system that meets OctopusMind expectation by enriching the database by extracting useful information from raw documents, namely, surface information for construction projects, renewal details allowing to anticipate in advance future projects and also financial information that can be used to bet-

ter filter relevant documents. Such extracted information is planned to be used in more complex tasks such as market analysis. Aside from information extraction we have also cleared the foundation for the creation of a recommendation system that suits the public procurement business. Thanks to our contribution in document embedding we are able to represent documents in such a way that the semantics is preserved.

Despite the pragmatic solutions we have been able to propose, many perspectives arise from our work. Especially in the multilingual domain. Indeed, within the time span available for addressing our different tasks, we train a model for each language. We are considering extending these models to work with different languages simultaneously. One interesting way of doing it, is to create a universal document embedding method that allows mapping texts in different languages in the same space while preserving their semantics. This may allow document recommendation to be proposed in multiple languages.

With the increasing amount of information we are able to extract and the ability to better represent the documents semantics, it becomes feasible to achieve large scale market analysis. This may be partially out the scope of this thesis, but still worth mentioning. As the NLP allowed us in some cases to go from texts to quantitative features (financial indicators) the combined use of these solutions allow us to handle the public market characterization within a bigger picture. It will bring a better understanding of the relationships and interactions between the different actors.

BIBLIOGRAPHY

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, « A neural probabilistic language model », *Journal of machine learning research*, vol. 3, *Feb*, pp. 1137–1155, 2003.

[2] R. Collobert and J. Weston, « A unified architecture for natural language processing: Deep neural networks with multitask learning », *in Proceedings of the 25th international conference on Machine learning*, ACM, 2008, pp. 160–167.

[3] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, « Distributed representations of words and phrases and their compositionality », *in Advances in neural information processing systems*, 2013, pp. 3111–3119.

[4] I. Sutskever, O. Vinyals, and Q. V. Le, « Sequence to sequence learning with neural networks », *in Advances in neural information processing systems*, 2014, pp. 3104–3112.

[5] D. Bahdanau, K. Cho, and Y. Bengio, « Neural machine translation by jointly learning to align and translate », *arXiv preprint arXiv:1409.0473*, 2014.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, « BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding », *in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[7] O. Ahmia, N. Béchet, and P.-F. Marteau, « Two Multilingual Corpora Extracted from the Tenders Electronic Daily for Machine Learning and Machine Translation Applications », *in Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[8] O. Ahmia, N. Béchet, P.-F. Marteau, and A. Garel, « Utilité d'un couplage entre Word2Vec et une analyse sémantique latente: expérimentation en catégorisation de données textuelles. », *in Extraction et Gestion des Connaissances: Actes de la conférence EGC'2019*, BoD-Books on Demand, vol. 79, 2019.

[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, « Learning phrase representations using RNN encoder-decoder for statistical machine translation », *arXiv preprint arXiv:1406.1078*, 2014.

[10] Y. Le Cun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard, « Handwritten digit recognition: Applications of neural network chips and automatic learning », *IEEE Communications Magazine*, vol. 27, *11*, pp. 41–46, 1989.

[11] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, « Hierarchical attention networks for document classification », *in Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016, pp. 1480–1489.

[12] J. S. Olsson, D. W. Oard, and J. Hajič, « Cross-language text classification », *in Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, 2005, pp. 645–646.

[13] L. Breiman, « Random forests », *Machine learning*, vol. 45, *1*, pp. 5–32, 2001.

[14] C. Cortes and V. Vapnik, « Support vector machine », *Machine learning*, vol. 20, *3*, pp. 273–297, 1995.

[15] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, « On combining classifiers », *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, *3*, pp. 226–239, 1998.

[16] N. Kushmerick, E. Johnston, and S. McGuinness, « Information Extraction By Text Classification », *in In The IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, 2001.

[17] L. Dey and S. M. Haque, « Opinion mining from noisy text data », *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 12, *3*, pp. 205–226, Sep. 2009, ISSN: 1433-2825. (visited on 10/16/2018).

[18] N. Jindal and B. Liu, « Review spam detection », *in Proceedings of the 16th international conference on World Wide Web - WWW '07*, Banff, Alberta, Canada: ACM Press, 2007, p. 1189, ISBN: 978-1-59593-654-7. (visited on 10/16/2018).

[19] Z. S. Harris, « Distributional Structure », *WORD*, vol. 10, *2-3*, pp. 146–162, Aug. 1954, ISSN: 0043-7956, 2373-5112. (visited on 10/16/2018).

[20] G. Salton, A. Wong, and C.-S. Yang, « A vector space model for automatic indexing », *Communications of the ACM*, vol. 18, *11*, pp. 613–620, 1975.

[21] K. S. Jones, « A statistical interpretation of term specificity and its application in retrieval », *Journal of Documentation*, vol. 28, pp. 11–21, 1972.

[22] T. K. Landauer, P. W. Foltz, and D. Laham, « An introduction to latent semantic analysis », *Discourse Processes*, vol. 25, *2-3*, pp. 259–284, Jan. 1998, ISSN: 0163-853X, 1532-6950. (visited on 10/16/2018).

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, « Efficient Estimation of Word Representations in Vector Space », *arXiv:1301.3781 [cs]*, Jan. 2013, arXiv: 1301.3781.

[24] J. Pennington, R. Socher, and C. Manning, « Glove: Global vectors for word representation », *in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[25] Q. V. Le and T. Mikolov, « Distributed Representations of Sentences and Documents », *arXiv:1405.4053 [cs]*, May 2014, arXiv: 1405.4053.

[26] A. M. Dai, C. Olah, and Q. V. Le, « Document embedding with paragraph vectors », *arXiv preprint arXiv:1507.07998*, 2015.

[27] G. Mesnil, T. Mikolov, M. Ranzato, and Y. Bengio, « Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews », *arXiv preprint arXiv:1412.5335*, 2014.

[28] J. H. Lau and T. Baldwin, « An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation », *in Proceedings of the 1st Workshop on Representation Learning for NLP*, 2016, pp. 78–86.

[29] Z. Zhu and J. Hu, « Context aware document embedding », *arXiv preprint arXiv:1707.01521*, 2017.

[30] M. Chen, « Efficient verctor representation for documents through corruption », *arXiv preprint arXiv:1707.02377*, p. 13, 2017.

[31] R. Ju, P. Zhou, C. H. Li, and L. Liu, « An Efficient Method for Document Categorization Based on Word2vec and Latent Semantic Analysis », *in 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, LIVERPOOL, United Kingdom, Oct. 2015, pp. 2276–2283, ISBN: 978-1-5090-0154-5.

[32] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, « Object recognition with gradient-based learning », *in Shape, contour and grouping in computer vision*, Springer, 1999, pp. 319–345.

[33] D. M. Blei, A. Y. Ng, and M. I. Jordan, « Latent dirichlet allocation », *Journal of machine Learning research*, vol. 3, *Jan*, pp. 993–1022, 2003.

[34] C. E. Moody, « Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec », *arXiv:1605.02019 [cs]*, May 2016, arXiv: 1605.02019.

[35] F. Rosenblatt, « The perceptron: a probabilistic model for information storage and organization in the brain. », *Psychological review*, vol. 65, *6*, p. 386, 1958.

[36] M. Minsky and S. Papert, « An introduction to computational geometry », *Cambridge tiass., HIT*, 1969.

[37] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, « Learning representations by back-propagating errors », *Cognitive modeling*, vol. 5, *3*, p. 1, 1988.

[38] K. Vora and S. B. Yagnik, « A Survey on Backpropagation Algorithms for Feed-forward Neural Networks », *International Journal of Engineering Development and Research (IJEDR)*, vol. 1, *3*, pp. 193–197, 2014.

[39] D. H. Ballard, « Modular Learning in Neural Networks. », *in AAAI*, 1987, pp. 279–284.

[40] A. Zell, *Simulation neuronaler netze*. Addison-Wesley Bonn, 1994, vol. 1.

[41] N. Halko, P.-G. Martinsson, and J. A. Tropp, « Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions », *SIAM review*, vol. 53, *2*, pp. 217–288, 2011.

[42] K. Fukushima, « Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position », *Biological cybernetics*, vol. 36, *4*, pp. 193–202, 1980.

[43] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, « Gradient-based learning applied to document recognition », *Proceedings of the IEEE*, vol. 86, *11*, pp. 2278–2324, 1998.

[44] T. Mikolov, K. Chen, G. Corrado, and J. Dean, « Efficient estimation of word representations in vector space », *arXiv preprint arXiv:1301.3781*, 2013.

[45] V. Nair and G. E. Hinton, « Rectified linear units improve restricted boltzmann machines », *in Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[46] Y. Bengio, P. Simard, P. Frasconi, *et al.*, « Learning long-term dependencies with gradient descent is difficult », *IEEE transactions on neural networks*, vol. 5, *2*, pp. 157–166, 1994.

[47] S. Hochreiter and J. Schmidhuber, « Long Short-Term Memory », *Neural Comput.*, vol. 9, *8*, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, « Attention is all you need », *in Advances in neural information processing systems*, 2017, pp. 5998–6008.

[49] J. Cheng, L. Dong, and M. Lapata, « Long Short-Term Memory-Networks for Machine Reading », *in Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 551–561.

[50] J. L. Ba, J. R. Kiros, and G. E. Hinton, « Layer normalization », *arXiv preprint arXiv:1607.06450*, 2016.

[51] Y. Gal and Z. Ghahramani, « A theoretically grounded application of dropout in recurrent neural networks », *in Advances in neural information processing systems*, 2016, pp. 1019–1027.

[52] S. Sukhbaatar, J. Weston, R. Fergus, *et al.*, « End-to-end memory networks », *in Advances in neural information processing systems*, 2015, pp. 2440–2448.

[53] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher, « Ask me anything: Dynamic memory networks for natural language processing », *in International conference on machine learning*, 2016, pp. 1378–1387.

[54] M.-T. Luong, H. Pham, and C. D. Manning, « Effective approaches to attention-based neural machine translation », *arXiv preprint arXiv:1508.04025*, 2015.

[55] T. Joachims, « A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization », *in Proceedings of the Fourteenth International Conference on Machine Learning*, ser. ICML '97, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 143–151, ISBN: 978-1-55860-486-5.

[56] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, « Rcv1: A new benchmark collection for text categorization research », *Journal of machine learning research*, vol. 5, *Apr*, pp. 361–397, 2004.

[57] W. Hersh, C. Buckley, T. Leone, and D. Hickam, « OHSUMED: an interactive retrieval evaluation and new large test collection for research », *in SIGIR'94*, Springer, 1994, pp. 192–201.

[58] E. Loper and S. Bird, « NLTK: the natural language toolkit », *arXiv preprint cs/0205028*, 2002.

[59] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, « Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1 », *in* D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds., Cambridge, MA, USA: MIT Press, 1986, ch. Learning Internal Representations by Error Propagation, pp. 318–362, ISBN: 0-262-68053-X. [Online]. Available: `http://dl.acm.org/citation.cfm?id=104279.104293`.

[60] T. Zhang, « Solving Large Scale Linear Prediction Problems Using Stochastic Gradient Descent Algorithms », *in Proceedings of the Twenty-first International Conference on Machine Learning*, ser. ICML '04, Banff, Alberta, Canada: ACM, 2004, pp. 116–, ISBN: 1-58113-838-5. DOI: `10.1145/1015330.1015332`. [Online]. Available: `http://doi.acm.org/10.1145/1015330.1015332`.

[61] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes, « Multinomial Naive Bayes for Text Categorization Revisited », *in Proceedings of the 17th Australian Joint Conference on Advances in Artificial Intelligence*, ser. AI'04, Cairns, Australia: Springer-Verlag, 2004, pp. 488–499, ISBN: 3-540-24059-4, 978-3-540-24059-4. DOI: `10.1007/978-3-540-30549-1_43`. [Online]. Available: `http://dx.doi.org/10.1007/978-3-540-30549-1_43`.

[62] D. J. Hand and K. Yu, « Idiot's Bayes—not so stupid after all? », *International statistical review*, vol. 69, *3*, pp. 385–398, 2001.

[63] J. MacQueen *et al.*, « Some methods for classification and analysis of multivariate observations », *in Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, vol. 1, 1967, pp. 281–297.

[64] A. K. Jain and R. C. Dubes, « Algorithms for clustering data », *Englewood Cliffs: Prentice Hall, 1988*, 1988.

[65] W. M. Rand, « Objective criteria for the evaluation of clustering methods », *Journal of the American Statistical association*, vol. 66, *336*, pp. 846–850, 1971.

[66] C. Laclau, F. d. A. T. de Carvalho, and M. Nadif, « Fuzzy co-clustering with automated variable weighting », *in 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Aug. 2015, pp. 1–8. DOI: 10.1109/FUZZ-IEEE.2015.7337802.

[67] J. C. Dunn, « A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters », *Journal of Cybernetics*, vol. 3, *3*, pp. 32–57, 1973. DOI: 10.1080/01969727308546046.

[68] A. Holzinger, « Human-Computer Interaction and Knowledge Discovery (HCI-KDD): What is the benefit of bringing those two fields to work together? », *in International Conference on Availability, Reliability, and Security*, Springer, 2013, pp. 319–328.

[69] S. C. de Abreu, T. L. Bonamigo, and R. Vieira, « A review on Relation Extraction with an eye on Portuguese », *Journal of the Brazilian Computer Society*, vol. 19, *4*, pp. 553–571, 2013.

[70] Z. S. Harris, « Distributional structure », *Word*, vol. 10, *2-3*, pp. 146–162, 1954.

[71] D. Ravichandran and E. Hovy, « Learning surface text patterns for a question answering system », *in Proceedings of the 40th annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2002, pp. 41–47.

[72] B. Min, S. Shi, R. Grishman, and C.-Y. Lin, « Ensemble Semantics for Large-scale Unsupervised Relation Extraction », *in Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, ser. EMNLP-CoNLL '12, Jeju Island, Korea: Association for Computational Linguistics, 2012, pp. 1027–1037. [Online]. Available: http://dl.acm.org/citation.cfm?id=2390948.2391062.

[73] D. Lin and P. Pantel, « DIRT@ SBT@ discovery of inference rules from text », *in Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2001, pp. 323–328.

[74] D. Beeferman and A. Berger, « Agglomerative clustering of a search engine query log », *in Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2000, pp. 407–416.

[75] L. B. Feldman, *Morphological aspects of language processing*. Psychology Press, 2013.

[76] D. Zelenko, C. Aone, and A. Richardella, « Kernel methods for relation extraction », *Journal of machine learning research*, vol. 3, *Feb*, pp. 1083–1106, 2003.

[77] L. Qian, G. Zhou, F. Kong, Q. Zhu, and P. Qian, « Exploiting constituent dependencies for tree kernel-based semantic relation extraction », *in Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, Association for Computational Linguistics, 2008, pp. 697–704.

[78] G. Zhou, L. Qian, and J. Fan, « Tree kernel-based semantic relation extraction with rich syntactic and semantic information », *Information Sciences*, vol. 180, *8*, pp. 1313–1325, 2010.

[79] S. Miller, H. Fox, L. Ramshaw, and R. Weischedel, « A Novel Use of Statistical Parsing to Extract Information from Text », *in Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, ser. NAACL 2000, Seattle, Washington: Association for Computational Linguistics, 2000, pp. 226–233. [Online]. Available: http://dl.acm.org/citation.cfm?id=974305.974335.

[80] M. Banko, O. Etzioni, and T. Center, « The Tradeoffs Between Open and Traditional Relation Extraction. », *in ACL*, vol. 8, 2008, pp. 28–36.

[81] G. Angeli, J. Tibshirani, J. Wu, and C. D. Manning, « Combining Distant and Partial Supervision for Relation Extraction. », *in EMNLP*, 2014, pp. 1556–1567.

[82] L. Wang, Z. Cao, G. De Melo, and Z. Liu, « Relation classification via multi-level attention cnns », *in Proceedings of the 54th annual meeting of the Association for Computational Linguistics (volume 1: long papers)*, 2016, pp. 1298–1307.

[83] Y. Xu, L. Mou, G. Li, Y. Chen, H. Peng, and Z. Jin, « Classifying relations via long short term memory networks along shortest dependency paths », *in proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 1785–1794.

[84] J. Lafferty, A. McCallum, and F. Pereira, « Conditional random fields: Probabilistic models for segmenting and labeling sequence data », *in Proceedings of the eighteenth international conference on machine learning, ICML*, vol. 1, 2001, pp. 282–289.

[85] N. Ye, W. S. Lee, H. L. Chieu, and D. Wu, « Conditional Random Fields with High-Order Features for Sequence Labeling », *in Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds., Curran Associates, Inc., 2009, pp. 2196–2204. [Online]. Available: `http://papers.nips.cc/paper/3815-conditional-random-fields-with-high-order-features-for-sequence-labeling.pdf`.

[86] S. R. Eddy, « Hidden markov models », *Current opinion in structural biology*, vol. 6, *3*, pp. 361–365, 1996.

[87] C. Sutton, A. McCallum, *et al.*, « An introduction to conditional random fields », *Foundations and Trends® in Machine Learning*, vol. 4, *4*, pp. 267–373, 2012.

[88] M. Collins, « Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms », *in Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, Association for Computational Linguistics, 2002, pp. 1–8.

[89] Y. Freund and R. E. Schapire, « Large margin classification using the perceptron algorithm », *Machine learning*, vol. 37, *3*, pp. 277–296, 1999.

[90] G. D. Forney, « The viterbi algorithm », *Proceedings of the IEEE*, vol. 61, *3*, pp. 268–278, 1973.

[91] Y. Li, J. Jiang, H. L. Chieu, and K. M. A. Chai, « IJCNLP », 2011, pp. 392–400.

[92] T. Lavergne, O. Cappé, and F. Yvon, « Practical Very Large Scale CRFs », *in Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ser. ACL '10, Uppsala, Sweden: Association for Computational Linguistics, 2010, pp. 504–513. [Online]. Available: `http://dl.acm.org/citation.cfm?id=1858681.1858733`.

[93] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, « Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization », *ACM Transactions on Mathematical Software (TOMS)*, vol. 23, *4*, pp. 550–560, 1997.

[94] M. F. Møller, « A scaled conjugate gradient algorithm for fast supervised learning », *Neural networks*, vol. 6, *4*, pp. 525–533, 1993.

[95] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, « Algorithms for hyper-parameter optimization », *in Advances in Neural Information Processing Systems*, 2011, pp. 2546–2554.

[96] H. I. Ansoff, « Managing strategic surprise by response to weak signals », *California management review*, vol. 18, *2*, pp. 21–33, 1975.

[97] N. Okazaki, « Crfsuite: a fast implementation of conditional random fields (crfs) », 2007.

[98] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, « Scikit-learn: Machine Learning in Python », *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[99] H. Chen, X. Li, and Z. Huang, « Link prediction approach to collaborative filtering », *in Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'05)*, IEEE, 2005, pp. 141–142.

[100] P. Brusilovsky and E. Millán, « User models for adaptive hypermedia and adaptive educational systems », *in The adaptive web*, Springer, 2007, pp. 3–53.

[101] K. Gábor, H. Zargayouna, I. Tellier, D. Buscaldi, and T. Charnois, « Unsupervised relation extraction in specialized corpora using sequence mining », *in International Symposium on Intelligent Data Analysis*, Springer, 2016, pp. 237–248.

[102] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, « Squad: 100,000+ questions for machine comprehension of text », *arXiv preprint arXiv:1606.05250*, 2016.

[103] B. McCann, N. S. Keskar, C. Xiong, and R. Socher, « The natural language decathlon: Multitask learning as question answering », *arXiv preprint arXiv:1806.08730*, 2018.

[104] W. L. Taylor, « “Cloze procedure”: A new tool for measuring readability », *Journalism Bulletin*, vol. 30, *4*, pp. 415–433, 1953.

[105] D. M. Blei, A. Y. Ng, and M. I. Jordan, « Latent Dirichlet Allocation », *Journal of Machine Learning Research*, vol. 3, *Jan*, pp. 993–1022, 2003, ISSN: ISSN 1533-7928.

[106] C. Cortes and V. Vapnik, « Support-vector networks », *Machine Learning*, vol. 20, *3*, pp. 273–297, Sep. 1995, ISSN: 0885-6125, 1573-0565.

[107] L. Rokach and O. Maimon, « Data mining and knowledge discovery handbook », *in* Springer, 2005, pp. 321–352.

[108] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, *et al.*, *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*, 2001.

[109] S. Sarawagi, « Information extraction », *Foundations and trends in databases*, vol. 1, *3*, pp. 261–377, 2008.

[110] D. Zeng, K. Liu, S. Lai, G. Zhou, J. Zhao, *et al.*, « Relation Classification via Convolutional Deep Neural Network. », *in COLING*, 2014, pp. 2335–2344.

[111] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, « Distant Supervision for Relation Extraction Without Labeled Data », *in Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ser. ACL ’09, Suntec, Singapore: Association for Computational Linguistics, 2009, pp. 1003–1011, ISBN: 978-1-932432-46-6. [Online]. Available: `http://dl.acm.org/citation.cfm?id=1690219.1690287`.

[112] V. N. Vapnik and V. Vapnik, *Statistical learning theory*. Wiley New York, 1998, vol. 1.

[113] O. Ahmia, N. Béchet, and P.-F. Marteau, « Apprentissage de structures séquentielles pour l’extraction d’entités et de relations dans des textes d’appels d’offres. », *in EGC*, 2017, pp. 351–356.

Titre : Veille stratégique assistée sur des bases de données d'appels d'offres par traitement automatique de la langue naturelle, fouille de textes et apprentissage profond.

Mot clés : Fouille de textes, Apprentissage profond, Attention hiérarchique, TALN, Appel d'offres, Apprentissage automatique

Résumé : Le data mining ou text mining est un concept récent apparu en 1989 qui engendre des innovations dans de nombreux domaines (knowledge management, marketing, communication, juridique, banque, finance, assurance, santé...). Face à la croissance des données disponibles, l'analyse humaine doit être accompagnée par des technologies de traitement automatique du langage naturel à la fois rapides, économes en ressource et de grande qualité pour servir d'interface entre l'homme et la machine.

Cette croissance exponentielle de l'information disponible en ligne génère une pression de plus en plus forte sur les algorithmes et technologies de traitement de cette information qui se présente principalement sous une forme textuelle. La nécessité de développer des méthodes avancées de TALN (traitement automatique du langage naturel) pour trouver, filtrer et analyser ces ressources de manière rapide et efficace devient ainsi d'autant plus prégnante.

C'est dans cette optique, que j'ai souhaité réaliser ma thèse sur des données économiques textuels que sont les avis d'appels publics à la concurrence, communément appelés appels d'offres. Le contenu de ces données est textuel, multilingue, peu ou pas structuré, et très hétérogène, tant dans sa forme que sur le fond. En effet, le contenu des marchés mélange à la fois du formalisme et le détail de l'intention d'achat. La capitalisation de ces documents sur plusieurs années a permis de disposer d'une banque de données qui constitue une mine d'informations dans différents domaines (secteurs d'activités, concurrence, prix d'achats...) permet d'asseoir une veille stratégique et concurrentielle sur un large spectre de métiers, de technologies ou de services.

Cette thèse, effectuée dans le cadre d'un contrat CIFRE avec la société OctopusMind, est centrée sur le développement d'un outillage informatique dédié et optimisé pour l'assistance à l'exploitation de la base d'appels d'offres, dans une finalité de veille stratégique. Cet outillage est basé sur les processus avancés de traitement de l'information et a pour vocation de s'intégrer dans le système d'information développé par l'entreprise. Les objectifs principaux de la thèse concernent la recommandation d'appels d'offres pour les clients, la classification des appels d'offres, ainsi que l'extraction d'informations pertinentes au regard d'une spécification de recherche donnée. Notre contribution se décline en trois chapitres : le premier concerne le développement d'une ressource multilingue partiellement comparable. Celle-ci est construite à partir des appels d'offres européens publiés par le TED (Tenders Electronic Daily). Elle contient plus de deux millions de documents traduits dans 24 langues publiées durant les 9 dernières années (cette ressource est une donnée ouverte). Le deuxième chapitre concerne une étude sur les questions de vectorisation de mots, phrases et documents susceptibles de capturer au mieux, des éléments d'ordre sémantique, selon différentes échelles. Nous avons proposé deux approches : la première est basée sur une combinaison entre un plongement de mot (word2vec) et une caractérisation de type sémantique latente (LSA). La deuxième est basée sur une architecture neu-

ronale originale basée sur des réseaux d'attention convolutionnels à deux niveaux. Ces vectorisations sont exploitées à titre de validation sur des tâches de classification et de clustering de textes. Le troisième chapitre concerne l'extraction de relations sémantiques contenues dans les appels d'offres, en particulier, permettant de relier des bâtiments à des surfaces, des lots à des budgets, etc. Les approches supervisées développées sont ici plus traditionnelles et reposent sur des Conditionnal Random Fields. La fin de ce chapitre aborde la mise en production dans l'environnement logiciel d'OctopusMind des différentes solutions développées, notamment l'extraction d'informations, le système de recommandation, ainsi que la combinaison de ces différents modules pour résoudre des problèmes plus complexes (i.e. études de marchés).

Title: Assisted strategic monitoring on call for tender databases using natural language processing, text mining and deep learning

Keywords: Text mining, Deep learning, Hierarchical attention, NLP, Call for tender, Machine learning

Abstract:

Data mining or text mining is a recent concept which appeared in 1989, it has created several innovations in many fields (knowledge management, marketing, communication, legal, banking, finance, insurance, health ...). Faced with the growth of available data, human analysis must go along with automatic natural language processing technologies that are both fast, resource-efficient and of high quality to serve as an interface between man and machine.

This exponential growth in the information available online (which is mainly presented in textual form) generates increasing pressure on algorithms and technologies used for processing this information. The need to develop advanced NLP (natural language processing) methods to find, filter and analyze these resources quickly and efficiently becomes all the more significant.

It is with this in mind that I wanted to carry out my thesis on textual economic data, more precisely of public calls for competition notices, commonly called calls for tenders. The content of this data is textual, multilingual, loosely structured or not at all, and very heterogeneous, both in form and content. Indeed, the content of the documents mixes both formalism and the detail of the purchase intention. The capitalization of these documents over several years has made it possible to have a database that constitutes a wealth of information in different fields (business sectors, competition, purchase prices, etc.). strategic and competitive across a broad spectrum of professions, technologies or services.

This thesis, carried out within the framework of a CIFRE contract with the OctopusMind company, is focused on developing a set of automated tools dedicated and optimized to assist call for tender databases processing, for the purpose of strategic intelligence monitoring. This set of tools is based on advanced information processing techniques and is designed to be integrated into the information system developed by the OctopusMind company. The main objectives of this thesis are centred on recommendation of calls for tenders for OctopusMind's clients, the classification of calls for tenders documents, as well as the extraction of relevant information with regard to a given set of specifications.

Our contribution is divided into three chapters: The first chapter is about developing a partially comparable multilingual corpus, built from the European calls for tender published by TED (Tenders Electronic Daily). It con-

tains more than 2 million documents translated into 24 languages published over the last 9 years (this resource is published as open data). The second chapter presents a study on the questions of words, sentences and documents embedding, likely to capture semantic features at different scales. We proposed two approaches: the first one is based on a combination between a word embedding (word2vec) and latent semantic analysis (LSA). The second one is based on a novel artificial neural network architecture based on two-level convolutional attention mechanisms. These embedding methods are evaluated on classification and text clustering tasks. The third chapter concerns the extraction of semantic relationships in calls for tenders, in particular, allowing to link buildings to areas, lots to budgets, and so on. The supervised approaches developed in this part of the thesis are essentially based on Conditionnal Random Fields. The end of the third chapter concerns the application aspect, in particular with the implementation of some solutions deployed within OctopusMind's software environment, including information extraction, a recommender system, as well as the combination of these different modules to solve some more complex problems, e.g. market analysis.